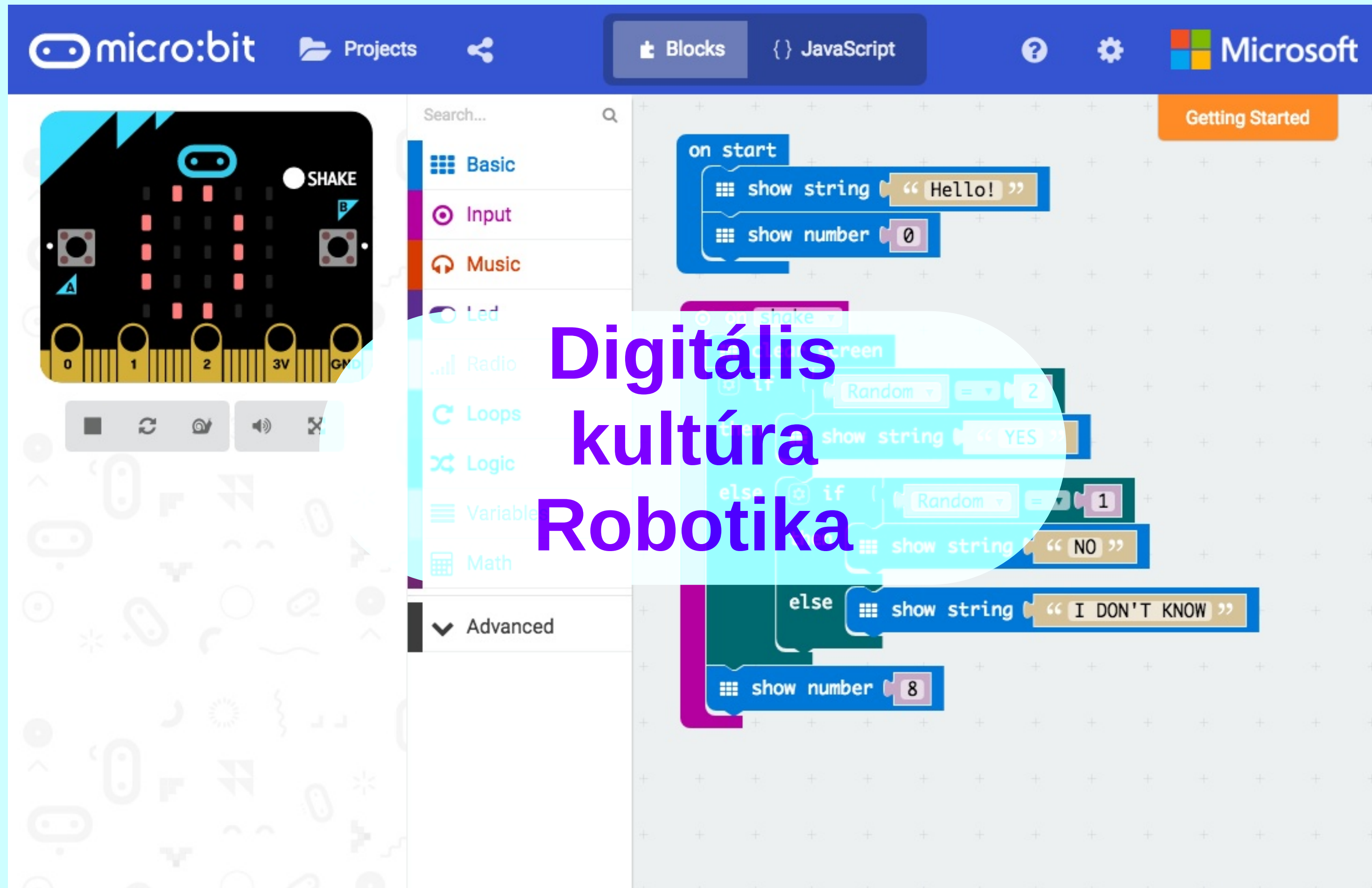




1. A számítástechnika szaktantermekben tanulóknak tartózkodni csak szaktanári engedéllyel lehet.
2. A géptermek szünetekben, használaton kívül zárva tartandók, csak az arra felhatalmazást kapott személyek nyithatják ki.
3. Internethasználat, tanórán kívüli gyakorlás csak felügyelet mellett történhet.
4. Az internetezési, levelezési etikett (Netikett) betartása mindenkire kötelező.
5. Digitális adathordozót csak szaktanári engedéllyel szabad a termekbe behozni.
6. Élelmiszert, rágógumit behozni tilos.
7. Mobiltelefont csak a tanár által meghatározott feladatra szabad használni.
8. A hálózati főkapcsolókat, a központi számítógépeket bekapcsolni; a központi gépekhez nyúlni tanulóknak tilos.
9. A számítástechnikai eszközök belsejébe nyúlni, azokat rongálni anyagi felelősség terhe mellett szigorúan tilos.
10. A számítógépeket bekapcsolni csak a szaktanár engedélyével, az előírásoknak megfelelő módon szabad.
11. Csak a tanár által meghatározott szoftverek használhatók.
12. A szoftverek beállításainak megváltoztatása szigorúan tilos!
13. Ha a foglalkozás ideje alatt a tanuló rendellenességet, a beállítások megváltoztatását, rongálás észlel, ha a számítógépe meghibásodik, azt azonnal köteles jelenteni, ellenkező esetben magára vállalja a felelősséget.
14. A géptermek rendjéért és tisztaságáért, az asztalok, monitorok, gépzázak, billentyűzetek, hangfalak (fülhallhatók), egerek állapotáért minden tanuló egyénileg is felelős.
15. Minden tanuló csak a saját helyén ülhet, melyet az ülésrendbe tanév elején be kell jegyezni.
16. A géptermi szabályzat mindenkire kötelező, aki bármelyik terembe belép!
17. Amennyiben valaki a géptermi munkarendet megsérti és ezzel anyagi kárt okoz – akár gondatlanul, akár szándékosan –, a szülő (törvényes képviselő) köteles a kárt a vonatkozó jogszabályok rendelkezése szerint megtéríteni.

Megjegyzések:

A géptermi szabályzat a 21. és 22. terem használatára egyaránt vonatkozik!

7. pont: Alapértelmezésben a termekben mobiltelefont használni tilos. Ez alól kivételt jelentenek azok a feladatok, amelyek során a mobiltelefon használata szükséges a feladat megoldásához. Ezen felül használható mobiltelefon vagy tablet az elektronikus segédanyagok (mint ez a dokumentum) eléréshez is.

12. pont: A szoftverbeállítások megváltoztatásának a tilalma a Windows rendszerre érvényes. A Linux (Ubuntu) rendszer esetében minden felhasználó saját beállításokkal rendelkezik, így azok szabadon megváltoztathatóak. Azonban a hálózati beállításokat itt sem szabad módosítani, mert az a gép működőképességére lehet hatással!

8. pont: A hálózati főkapcsolókra vonatkozó tilalmat szükség esetén felülírhatja, ha balaset- vagy tűzvédelmi okokból szükséges a terem azonnali áramtalanítása.



A robotika a számítástechnika egy interdiszciplináris területe. Beletartozik a robotok tervezése, építése, működtetése és használata is. A robotika célja intelligens gépek tervezése, amelyek az emberek mindennapi életét képesek valamilyen módon segíteni biztonságosabbá téve életünket. A robotika több tudományág eredményeit is felhasználja: informatika, számítástechnika, mechanikai és elektromos mérnöki területek többek között.

A robotika olyan gépeket fejleszt, amelyek helyettesíthetik az embereket azok tevékenységének elvégzése révén. Ilyen módon az emberek veszélyes, illetve monoton feladatok elvégzésétől mentesülhetnek, tevékenységüket inkább az emberi gondolkodást, kreativitást jobban igénylő területekre koncentrálhatják. Robotok alkalmazhatók rengeteg helyen, beleértve akár kifejezetten veszélyes területeket is. Például éltveszélyes környezetben: radioaktív vagy mérgező környezetben. Robotokkal végezhető akár bombák felkutatása, hatástalanítása is. Bejuttathatók olyan helyekre, amelyek méretük vagy statikai instabilitás



Karel Čapek (1890 – 1938)

A kép forrása: Wikipédia
(https://en.wikipedia.org/wiki/Karel_%C4%8Capek)

miatt emberi jelenlétre alkalmatlanok (például egy összeomlott épület belsejében túlélők után kutathatnak). Dolgozhatnak robotok olyan helyen, ahol emberek túlélése nem biztosítható: világűrben, vízalatti környezetben, más bolygók felszínén stb.

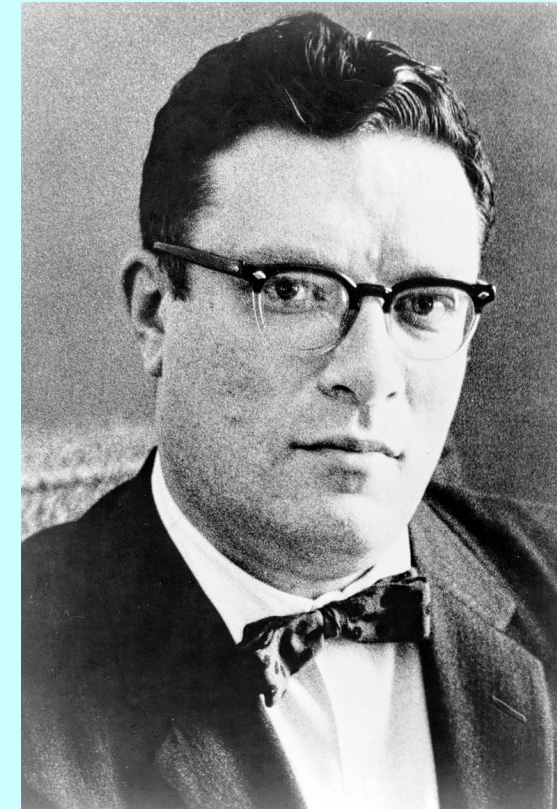
Egy robot hasonlíthat is akár az emberre vagy egyéb élőlényre, de ez nem szükségszerű. Az iparban például olyan robotokat alkalmaznak, amelyeknek egyáltalán nincs emberi kinézetük, ám alakjuk, kialakításuk kifejezetten alkalmassá teszi a feladatuk elvégzésére.

A *robot* szó először a cseh **Karel Čapek** 1920-ban kiadott regényében merül fel. A robot szó a szláv *robota* szóból ered, amely *szolgát* jelent. A mű egy gyárban játszódik, amely robotnak nevezett mesterséges embereket készít, amelyek könnyen összetéveszthetők a valódi emberekkel – akárcsak a modern kori androidok gondolata előzménye is lehet. Valójában nem Karel Čapek írta le először a robot szót. Állítása szerint bátyja találta azt ki.

A *robotika* szó első előfordulása **Isaac Asimov** *Liar!* című novellájában jelent meg, amelyet 1941-ben publikáltak. Asimov műve megírásakor nem tudta, hogy a szót ő írta le először, amikor megalkotta a később rengeteg művében szereplő koncepciót a "Robotika három törvényéről".

A következőkben a robotika alapjait egy sokkal egyszerűbb eszközzel, a következő oldalon bemutatott kis eszközzel ismerheted meg. Ez az eszköz a BBC micro:bit nevű eszköze (jobboldali képen), amelyet kifejezetten oktatási célra fejlesztett ki a BBC (igen, ez a britek állami televíziója). Az eszköz köré akár egy robot is építhető, amelyet képes vezérelni. Mi most az alapvető irányítását fogjuk megismerni ennek az eszköznek a Robotika témakörön belül.

Az órák során Dr. Abonyi-Tóth Andor: Programozzunk micro:biteket! című művét használjuk útmutatásul. Az itt szereplő, forrásmegjelöléssel el nem látott ábrák, ha nem saját készítésűek, akkor ebből a könyvből származnak, ahogyan az ismertetett feladatok is.



Isaac Asimov (1920 – 1992)

Kép forrása: Wikipédia
(https://en.wikipedia.org/wiki/Isaac_Asimov)



A kép forrása: <https://thumbor.forbes.com>



Mindig tiszta asztalon dolgozz! Különösen veszélyesek az eszköz számára a rövidzárlatot okozó tárgyak, mint pl. gemkapocs, tűzőgép kapocs, körző, egy ceruza grafithegye, kifojt folyadék, alufóliadarab stb.

Az elektrosztatikus kisülés akár tönkre is teheti az eszközt, ezért fontos, hogy mielőtt az eszközhöz nyúlsz, a ruhádban (főleg a műszálasban!) összegyűlt töltést valahogy le kell vezetni. Erre alkalmas a számítógép házához vagy a radiátorhoz hozzáérés.

Az eszköz úgy legyen az asztalon elhelyezve, hogy ne lehessen véletlenül lesodorni, lelökni onnan.

Az egyes gesztusok kipróbálása során (pl. rázás) ügyelj arra, hogy biztosan tartsd a kezedben, nehogy kiessen belőle.

A tevékenységek során ügyelj arra, hogy se az eszközben, se a társaidban ne tegyél kárt.

Elemről csak addig működtesd az eszközt, amíg feltétlenül szükséges, egyébként használt az USB kábelt és ezáltal a számítógép lássa el árammal.

Ne köss az eszköz kivezetéseihez olyan külső eszközöket, amelyek nem kompatibilisek vele, mert például nagyobb tápfeszültséget igényelnek, mint amit az biztosítani tud! Ez amellet, hogy az eszközt tönkre teheti, az esetleges túlmelegedés tüzet vagy égési sérüléseket okozhat!

Az eszköz érzékelőit megzavarhatják külső tényezők: erős mágneses tér, fém tárgyak, stb. Ha a szenzorok nem a megfelelő értéket mérik, vizsgáld meg, nincs-e a környezetben valamilyen zavaró tényező, ami ezt az eltérést okozhatja.

1. Ismerkedés a micro:bit eszközzel

A BBC (a brit állami média) kifejezetten oktatási célra kifejlesztett, egy lapkás mikrovezérlője a micro:bit, amellyel meg fogunk ismerkedni. 4x5 cm méretű lapkáról van szó, amelyen egy 5x5-ös elrendezésű LED-mátrix található. Ennek a LED-mátrixnak az egyes elemeit ki lehet világítani, amellyel az eszköz kommunikálni tud a külvilággal.

De egyéb eszközei is vannak a kommunikációra, ahogyan az a jobboldali felső ábrán látható: gyorsulásérzékelővel, hőmérséklet érzékelővel, fényérzékelővel, irány érzékelő szenzorral is rendelkezik. Ezen felül be- és kimeneti csatlakozókkal, 2 nyomógommbal, bluetooth/rádió kapcsolódási lehetőséggel kommunikálni is képes. Nem csak a felhasználójával, hanem másik micro:bit lapkával is. Ezáltal több lapka képes együttműködni, amely további felhasználási lehetőségeket teremt.

Az eszköz képességei igazán sokrétűek lehetnek, így azok felsorolása nem is lehetséges.

Az eszközt a BBC ötlete alapján 29 partner bevonásával készítették.

A lapka 20 db NYÁK csatlakozó érintkezővel, 3 digitális/analóg gyűrűs csatlakozóval és 3 V-os kimeneti csatlakozóval rendelkezik, amelyekre számos eszközt lehet csatlakoztatni. Ilyen eszközök lehetnek például: szenzorok, szervó motorok, LED-ek, hangszórók stb. Így például a lapka, vagy több lapka együttműködve akár egy robot vezérlését is végezheti.

De mivel ez az eszköz csupán egy mikrovezérlő, így a felprogramozásához szükséges egy számítógép is. Ez a számítógép nem csak asztali gép lehet, hanem akár egy tablet vagy egy okostelefon is. Ezt a számítógépet USB kábellel vagy Bluetooth kapcsolaton keresztül kapcsolhatjuk össze az eszközzel.

A felprogramozás alatt azt értjük, hogy az eszköz által végrehajtandó programot egy számítógépen kell összeállítani, majd ezt kell feltölteni az eszközre, egészen pontosan annak a memóriájába, hogy ezt követően a processzora az utasításokat végrehajthassa.

A program többféle formában is elkészíthető a micro:bit számára: blokkszerkesztő segítségével vagy akár Python nyelven is lehetőségünk van elkészíteni a programot, amelyet azután az eszközre feltölthetünk.

Mi a következőkben a JavaScript Blokk Editor nevű, webes felületű blokkszerkesztőt fogjuk az eszköz programozására használni. A Scratch-ből már ismerős lesz a használata. A <http://makecode.microbit.org> címen elérhető felület látható a jobb oldali alsó ábrán. A PDF-ben a képre kattintva a böngésző-

be be is tölthető az oldal.

A következő oldalon levő ábra mutatja a szerkesztő felület részzeit részletesebben.

Az eszköztáron (középső, függőleges rész) az egyes elemek blokkok csoportját tartalmazzák. Rákattintva valamelyikre, kinyílik a tartalma, amiből bármelyik elemet megfoghatjuk az egérrel és a jobboldali munkaterületre ejthetjük. A munkaterületen levő elemet kiválasztva a Delete billentyűvel törölhetjük a legkönnyebben, de az egérrel megfogva a munkaterületből ki is dobhatjuk.

Jobb egér gombbal rákattintva a munkaterületen levő blokkra, a helyi menüt lehet megjeleníteni, ahol szintén van törlési lehetőség, de a blokkról másolat is készíthető, megjegyzés fűzhető hozzá, vagy a Súgó is előhívható (amely viszont nincs magyarra fordítva). Megfelelő angol tudás birtokában a Súgóban az egyes blokkokról hasznos információkhoz lehet jutni...

A munkaterületen a blokkokat egymáshoz kell kapcsolni, hogy működőképes programot kapjunk. Amíg a blokk nincs a megfelelő helyre kapcsolva, nem fog működni. Egy új projekt indításakor már eleve van a munkaterületen két kék színű blokk. Minden további blokknak ezek valamelyikéhez kell kapcsolódnia.

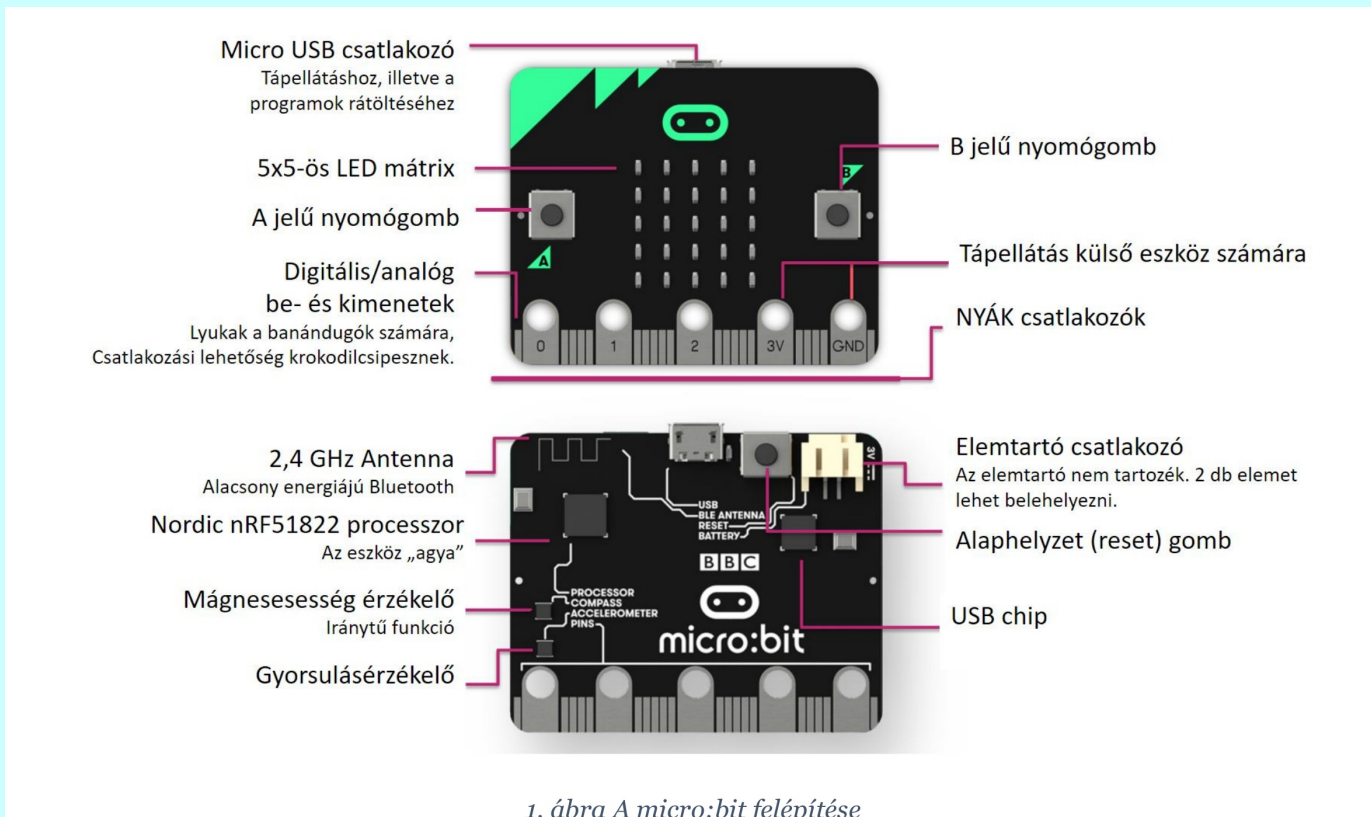
A blokkprogramozás nagy előnye, hogy az egyes blokkok kapcsolódási felülete eltérő, így nem lehet akármelyik blokkot akármelyikkel összekapcsolni, csak az egymásba illőket. Így elkerülhetjük azokat a szintaktikai hibának nevezett problémákat, amelyek a hagyományos programírás során sűrűn előfordulnak a nem összeillő elemek egymás mellé írása során.

Ez azonban még nem jelenti azt, hogy a programunk mindig helyesen fog működni! A számítógép nem gondolatolvasó, így nem tudja kitalálni, mit akarunk vele csináltatni! Amennyiben mást mondunk neki, mint amit szeretnénk, akkor mást fog csinálni! Egészen pontosan: a számítógép mindig pontosan azt teszi, amire utasítják, függetlenül attól, hogy az értelmes-e vagy sem!

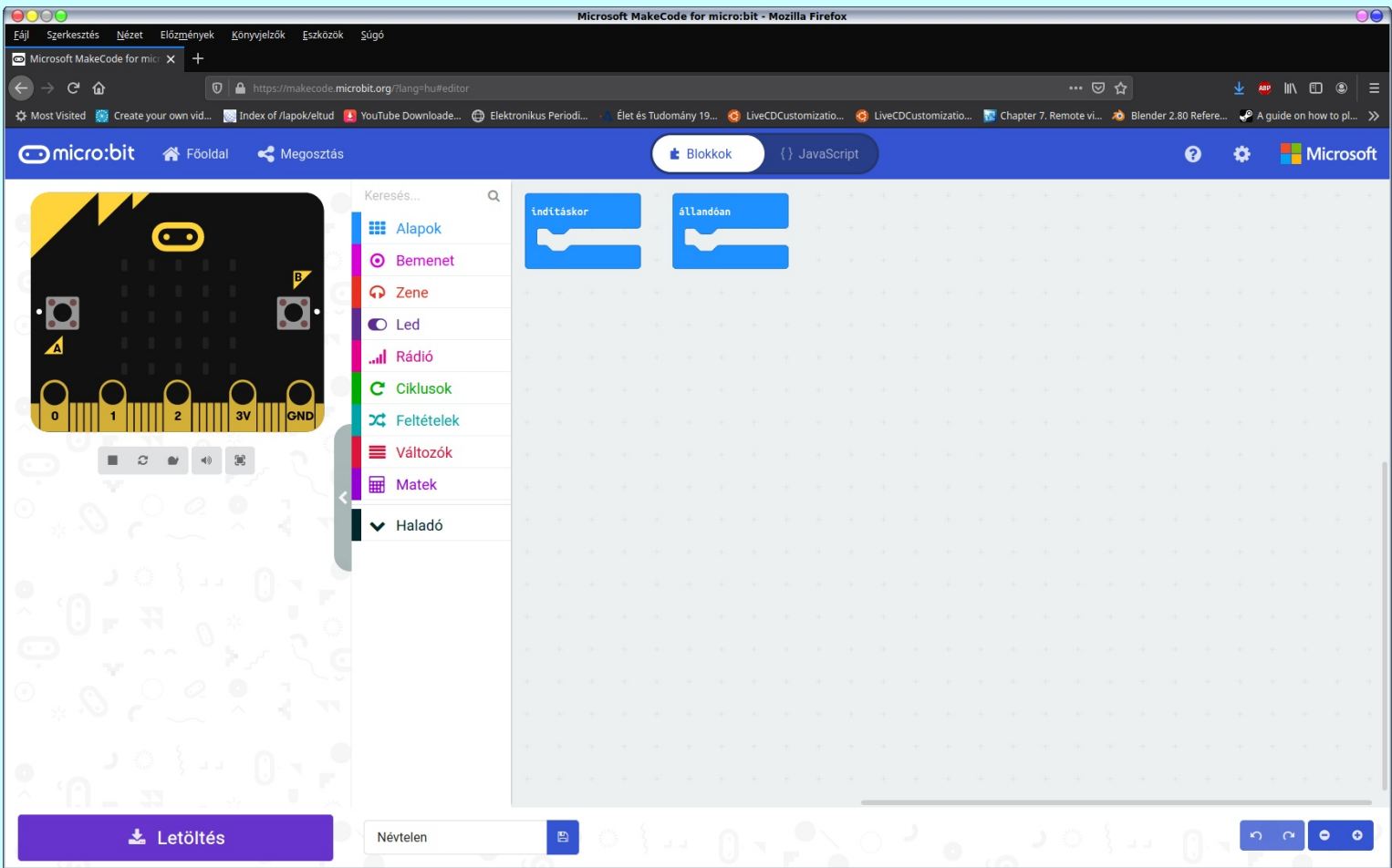
Ezért arra, hogy mire utasítjuk a számítógépet, nekünk kell odafigyelnünk.

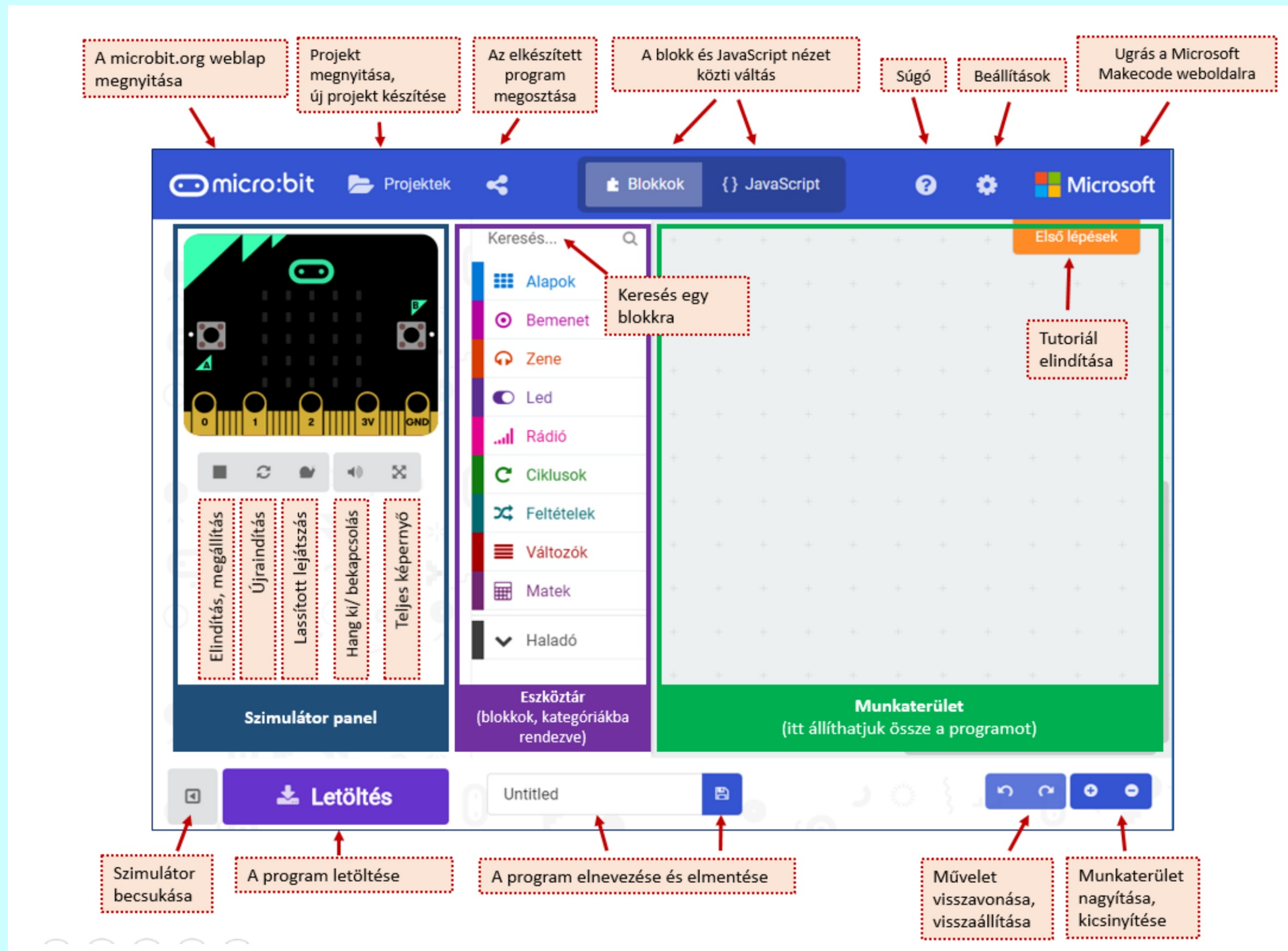
A jobboldalon vagy látszik, vagy nem a szimulátor, amely magát a micro:bit eszközt mutatja. Az eszköztár baloldalán levő szürke nyilas sáv segítségével lehet elrejteni vagy megjeleníteni a szimulátort.

A szimulátor azért hasznos, mert még mielőtt ténylegesen



1. ábra A micro:bit felépítése





feltöltenénk az eszközre a programot, ki lehet próbálni. Legalábbis a legtöbb lehetőséget ki lehet próbálni, de nyilván például a hőmérőt így nem lehet használni. Viszont a gombok lenyomását igen, hiszen a gombra rá lehet az egérrel kattintani. Igaz, itt sincs meg minden lehetőség, hiszen egy egérrel egyszerre nem lehet mindkét gombra kattintani, így azt a lehetőséget nem lehet szimulálni, amihez mindkét gombot egyszerre kellene megnyomni.

A szimulátor azokat a bemeneti eszközöket szimulálja, amelyek szerepelnek a programunkban. Tehát például ha a rázást mint bemenetet alkalmazza a programunk, akkor a szimulátoron megjelenik a rázás funkció is.

A szimulátor alatti gombok segítségével lehet (balról jobbra haladva)

1. a program futását elindítani vagy megállítani,

2. a programot futás közben újraindítani,

3. a program futását lassítani, hogy jobban meg lehessen figyelni annak folyamatát,

4. ki/be kapcsolni a hang lejátszását,

5. a szimulátort teljes képernyős módba állítani, vagy onnan visszahozni.

A teljes képernyős mód valójában csak a böngésző ablakán belülre vonatkozik, de így is jó lehetőség arra, hogy nagyobb méretben érjük el a szimuláció futása alatt a szimulátort.

Az oldal alján a szerkesztő tartalmaz még néhány gombot, amelyek akkor jönnek jól, amikor a program elkészült.

Amennyiben a tesztelés után a programot az eszközön is szeretnénk futtatni, akkor a Letöltés gombra kell a bal alsó sarokban kattintani. Ez a gomb tehát valójában nem letöltést végez, hanem az esz-

közre történő feltöltést. Ehhez annak a számítógépünkhöz USB kábelrel keresztül kell csatlakoznia.

Amennyiben a számítógépünkre szeretnénk a kész programot lementeni, akkor a Letöltés gomb jobb oldalán levő mentés funkciót kell használni. Először adjuk meg a programot tartalmazó állomány leendő nevét, majd a Mentés gombra kattintva történik meg annak letöltése.

Amennyiben a programot nem töltjük le, akkor a Főoldalon a saját projektek között bármikor megtalálhatjuk és újra megnyithatjuk. Ám az oldal alapból a gépünk alapján azonosít, így például az otthon készített projektet az iskolai gépen valószínűleg nem fogjuk látni, ahogyan akkor sem, ha az iskolában másik gépet használunk, mint amin a projekt készült! Az ebből eredő problémák kerülhetők el a program lementésével.

Röviden érdemes a blokkok fajtáit is megismerni, bár erre a következőkben még rengeteg időnk lesz! A munkaterületen alapból megtalálható két blokk egyike az „indításkor” (angolul: on start) blokk, amelybe helyezett blokkok a program indulásakor hajtódnak végre, a másik alapból elérhető blokk, az „állandóan” (angolul: forever) belsejében levő blokkok folyamatosan ismétlődnek. Attól függően tehát, hogy éppen mit akarunk az egyes blokkokkal, el kell döntenünk, melyik hova kerüljön.

Ha például az A gomb megnyomásakor szeretnénk valamilyen tevékenységet előírni a micro:bit számára, akkor a Bemenet kategóriában található „amikor a(z) A gombot lenyomva” (angolul: on button A pressed) blokkra van szükség (igen, hibásan van magyarra fordítva a blokk szövege). A B gombhoz ugyanerre a blokkra van szükség, de át kell állítani a legördülő

listában a használandó gombot. Amit ebbe a blokkba teszünk, annak a végrehajtása a kiválasztott gomb lenyomáskor fog megtörténni.

indításkor

állandóan

amikor a(z) A gomb lenyomva

Értelemszerűen az „indításkor” blokkba helyezett kód csak egyszer, a program indulásakor fog végrehajtódni. Ezért ide azt tesszük, amit minden más tevékenység előtt egyszer végre kell hajtani.

Az „állandóan” blokkba helyezett tartalom folyamatosan ismétlődik a háttérben, miközben egyéb tevékenységet végez a micro:bit eszköz. Ha például folyamatosan ismétlődő animációt vagy zenét akarunk, akkor annak blokkjait kell ide tenni.

A Bemenet (angolul: input) kategóriában található olyan vezérlőblokkok, amelyekben valamely esemény bekövetkezésekor végrehajtandó tevékenység helyezhető el. Ilyen esemény például, amikor az eszközön található valamelyik (A illetve B jelű) gombot vagy egyszerre mindkettőt benyomjuk. A blokk belsejében található egy legördülő lista, amelyben három lehetőség állítható be: A, B vagy az egyszerre mindkét gomb megnyomását jelentő A+B érték.

További bemeneti eseményekkel is meg fogunk ismerkedni a későbbiekben.

A fent felsorolt blokkok mindegyike olyan, amelynek csak a belsejébe helyezhető el másik blokk, sem elé, sem mögé nem. Ezek vezérlő blokkok, amelyek valamilyen esemény (jelen esetben a program indulása, a háttérben futó tervékenység, illetve egy gomb lenyomása) bekövetkezésekor végrehajtandó tevékenység megadására

show string

show number

show number

show number

“Visszaszámlálás indul”

3

2

1

szolgálnak. Az elvégzendő tevékenységet a vezérlőblokk belsejében kell elhelyezni.

A tényleges tevékenységet megadó blokkok ellenben egymás után kapcsolhatóak, jelezve ezzel az általuk leírt tevékenység egymást követő végrehajtási sorrendjét (a későbbiekben majd ezt fogjuk szekvenciális végrehajtásnak nevezni). Azonban ezek között a blokkok között is található olyan, amelynek a belsejébe is lehet további blokkokat elhelyezni, ahogy azt a későbbiekben látni fogjuk.

További lehetőség még a blokkok jobb oldalára illeszthető további blokk, amellyel az adott blokk paraméterezhető. Például az alaptevékenységet megadó blokk megadja, hogy például egy számot kell a LED-mátrix segítségével megjeleníteni. A jobboldalra pedig beilleszthető a megjelenítendő szám, vagy akár egy egész számítási művelet, amelyet végre kell hajtani a megjelenítendő szám meghatározásához.

show number

3 + 4 ÷ 2

Például a képen látható számításhoz szükséges blokkokat megtaláljuk a Matek kategóriában.

Térjünk vissza egy kicsit még a projektszerkesztő környezet felépítéséhez! A Főoldalra vezető link mellett szerepel egy „Megosztás” gomb is. Ez arra jó, hogy az elkészült programot másokkal megoszthassuk. Erre kattintva az alábbi képen látható ablakot kapjuk.

Projekt megosztása

Névtelen

Közzé kell tenned a projektet a megosztásához vagy más weblapokra való beágyazásához. Tudomásul veszed, hogy hozzájárulsz a projekt közzétételéhez.

Projekt közzététele

Mint a képen látható, adni kell neki egy nevet. Ez a név egyébként is hasznos, hiszen enélkül a Főoldalon is „Névtelen” elnevezéssel fog a projekt megjelenni. Ha nevet adtál a projektnek, akkor lehet a „Projekt közzététele” gombra kattintani. De ezzel még nem vagy kész!

Következő lépésként kapsz egy URL címet, amelyen a

projekt nyilvánosan elérhető. Ezt másold vágólapra és másold be oda, ahol közzé akarod tenni.

Jobboldalt felül található egy kis fogaskerék, ahol van még néhány lehetőség. Ezek közül a legelső, a Projekt beállítása is lehetőséget ad a projekt elnevezésére, így nem feltétlenül kell ahhoz publikussá tenni, hogy nevet adj neki.

A Kiterjesztések menüpont alatt a lehetőségeket lehet tovább bővíteni, amelyeket a micro:bit programozásánál lehet használni. Ezekre majd akkor lesz szükség, ha már az alpból elérhető lehetőségek kevésnek bizonyulnak.



Microsoft MakeCode for micro:bit - Mozilla Firefox

micro:bit

Főoldal

Megosztás

Blokkok

JavaScript

Keresés...

Alapok

Bemenet

Zene

Led

Rádió

Ciklusok

Feltételek

Változók

Matek

Haladó

show string

show number

show number

show number

“Visszaszámlálás indul”

3

2

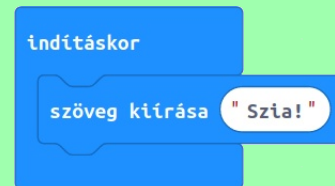
1

Letöltés

Névtelen

Szívdobbanás animáció

Készítsünk egy egyszerű animációt, amely azzal indul, hogy kiírja a „Szia!” szöveget, majd egy szív nagyobb és kisebb változatát változtatjuk, ezzel elérve, mintha dobogó szívet látnánk!



Az „indításkor” blokk tartalma a futás kezdetén hajtódik végre. Az ebben elhelyezett szövegkiíratással fog tehát kezdődni a program.

Mivel a szívdobogást folyamatosan ismételni akarjuk, ezért azt az „állandóan” blokkba kell elhelyezni. Ennek a blokknak a tartalma addig ismétlődik, amíg le nem állítjuk valahogyan a programot.

Ide kell elhelyezni két példányt az „ikon megjelenítése” blokból. Az egyikben egy kis méretű szívet, a másikban egy nagy méretű szívet kell az elérhető ikonok közül kiválasztani.

Szükség van továbbá némi késleltetésre is, hogy a szívdobbanás látható legyen. Erre szolgál a „szünet” blokk, amely a megadott ezredmásodpercig megállítja a végrehajtást.

Mielőtt az eszközre töltenéd, próbáld ki a szimulátorban az eredményt!

Minden szükséges blokkot az „Alapok” kategóriában találsz.



A szívdobbanás variálása

Nem csak szöveget, hanem számot is meg tud jeleníteni az eszköz. Ehhez a „szöveg kiírása” blokkot kell használni. A „Szia!” szöveg helyett az eszköz számoljon vissza háromtól (3, 2, 1)!

A szívdobogás helyett, a rendelkezésre álló ikonok felhasználásával készíts egy másik animációt!

Az elkészített animáció webcímét oszd meg a többiekkel a Google Drive felületen! Nézd meg, társaid milyen animációkat készítettek.



Integető robot

A LED-mátrix felett látható a micro:bit logója, amely olyan, mintha egy robot feje lenne. Használjuk ki ezt a lehetőséget, így csak a robot testét kell megrajzolnunk.

Készítsünk közösen olyan animációt, amelyben a robot folyamatosan integet a jobb kezével. Ekkor már nem előre megadott ikonokat használunk, hanem nekünk kell megadni, hogy a LED melyik pontjai világítsanak.

Ahhoz, hogy folyamatosan ismétlődjön az animáció, az „állandóan” blokkot kell használnunk.

Ebben kell elhelyezni a „ledek bekapcsolása” blokkot (még mindig az „Alapok” kategóriát használjuk), amelyben be kell kapcsolni a világító pontokat.

Kihasználva, hogy a blokkok másolhatók, a másolaton változtassuk meg a megfelelő LED-pontok bekapcsoltságát

Próbáld ki az animációt először szimulátorban, majd az eszközön!

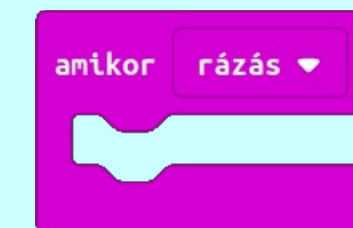
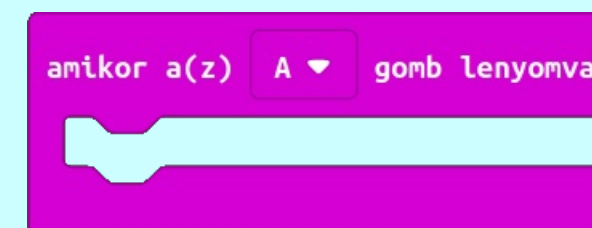


Fejlesszük tovább a projektet úgy, hogy

- indításkor egy álló alak látszódjon,
- az 'A' gomb hatására integessen a robot először egyet, majd oldjuk meg, hogy négyszer ismétlődjön az integetés,
- rázás hatására törlődjön le a képernyő.

Szükséges új blokkok:

(Kék: Alapok, Bordó: Bemenet, Zöld: Ciklusok kategóriából)



Integető robot továbbfejlesztve

Fejleszd tovább az alkalmazást úgy,

- 1. hogy a ,B’ gomb megnyomásakor a robot a bal karjával integessen, szintén négyszer, majd jelenjen meg az álló alak!
- 2. Ha az ,A’ és ,B’ gombot egyszerre nyomjuk meg, akkor a robot mindkét kezét emelje fel, majd engedje le!
- 3. Akkor is integessen a robot, ha balra vagy jobbra döntjük az eszközt! Ehhez ugyanaz a blokk kell, mint a rázáshoz, de más beállítással.
- 4. Találj ki egyedi robot alakot vagy animációt arra az esetre, ha az eszközt felfelé, illetve lefelé billentjük! („Logó fenn”, illetve „Logó lenn” események kellenek hozzá.)

Készíts új projektet!

Ha az eszközt

- balra döntjük, akkor jelenjen meg egy balra mutató nyíl (Nyugat – West)
- jobbra döntjük, akkor jelenjen meg egy jobbra mutató nyíl (Kelet – East)
- felfelé döntjük, akkor jelenjen meg egy felfelé mutató nyíl (Észak – North)
- lefelé döntjük, akkor jelenjen meg egy lefelé mutató nyíl (Dél – South)
- megrázzuk, legyen letörölve a képenyő.

A fentiekhez új blokkként kell az „Alap” kategóriából a további részbe elrejtett „nyíl megjelenítése” nevű blokkot használnod.

Azt is meg tudjuk különböztetni, hogy a ledes kijelző felfelé (mennyezet felé) vagy lefelé (padló felé) néz. Ehhez a rázás helyett a „képernyő fenn”, illetve „képernyő lenn” eseményt kell kiválasztani. Fejleszd tovább a projektet úgy, hogy ha lefelé néz a képernyő, akkor legyen kigyújtva a középső pont, ha felfelé, akkor pedig a sarokban levő négy pont.



Az ábrán látható események mind a „Bemenet” kategória „amikor” blokkjának legördülő listájában érhetőek el a rázás helyett.

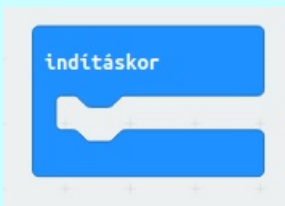


Most már tudod, hogyan írhatasz ki szöveget a kijelzőre, hogyan jeleníthetsz meg ikonokat, saját rajzokat. Megtanultad, hogyan kezelheted le a gombnyomásokat, a rázást, billentést és más gesztusokat. Ezen ismeretek felhasználásával készíts animációkat!

Az alábbiakban találsz néhány feladatot, de nyugodtan valósítsd meg saját ötleteidet! A projekteket mentsd el és oszd meg a többiekkel is.

- 1. Készíts egy mini animációs filmet. Indulhat a film szöveggel, utána pedig elmesélhetsz egy rövid történetet animációk segítségével!
- 2. Készíts olyan animációt, melynek szereplője egy tetszőleges állat. Az ,A’ és ,B’ gombok megnyomásával, valamint a különböző események során (pl. rázás, döntés) más-más animáció játszódjon le. (Pl. menjen balra, jobbra, ugorjon egyet stb.)
- 3. Készíts olyan animációt, amelyben a kijelző felső sorában 3 csillag (pont) látható. Öt másodperc elteltével zuhanjon le az egyik csillag, két másodperc múlva egy másik csillag, végül újabb két másodperc elteltével az utolsó csillag is.
- 4. Készíts olyan animációt, amelyben a lehulló esőcseppekből egy tócsa alakul ki.
- 5. Készíts egy időjárás előrejelző alkalmazást. Az ,A’ gomb hatására jelenjen meg egy napsugár, a ,B’ gomb hatására egy esőfelhő. A két gomb együttes megnyomásakor egy hópihe alak látszódjon. Rázáskor legyen letörölve a kijelző.

A megismert blokkok



A program tartalma a program indításakor fut le.



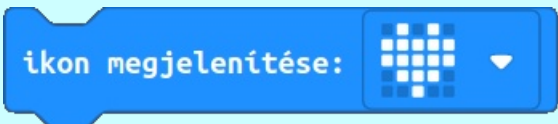
A blokk tartalma a háttérben állandóan ismétlődve fut.



A megadott szöveget kiírja a LED-mátrixra. A szöveg meghatározására további kifejezés-blokkok is használhatóak.



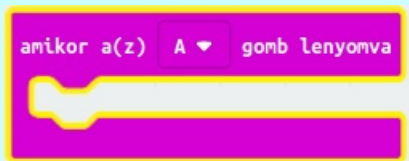
A megadott számot kiírja a LED-mátrixra. A szám meghatározására további kifejezés-blokkok is használhatóak.



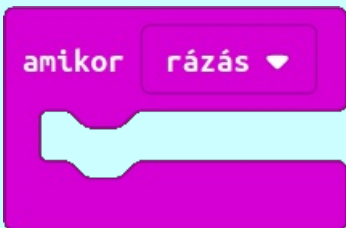
A kiválasztott ikon megjelenítése. A legördülő listából lehet az előre definiált ikonok közül választani.



Letörli a képernyőt, azaz a LED-mátrix valamennyi pontja kialszik.



Az **A**, **B** vagy egyszerre mindkét (**A+B**) megnyomásakor hajtódik végre a bele helyezett tartalom. Hogy melyik gomb megnyomására vonatkozzom az esemény, azt a benne található legördülő listában lehet beállítani.



A rázás, logó fent, logó lent, képernyő fent, képernyő lent, balra döntés, jobbra döntés, szabadesés, 3g, 6g, 8g események valamelyikének bekövetkezésekor végrehajtandó műveletek megadására szolgál. Hogy pontosan melyik eseményre hasson a fentiek közül, azt a legördülő listában kell beállítani.



A blokk tartalmát pontosan a megadott számú alkalommal ismétli meg.



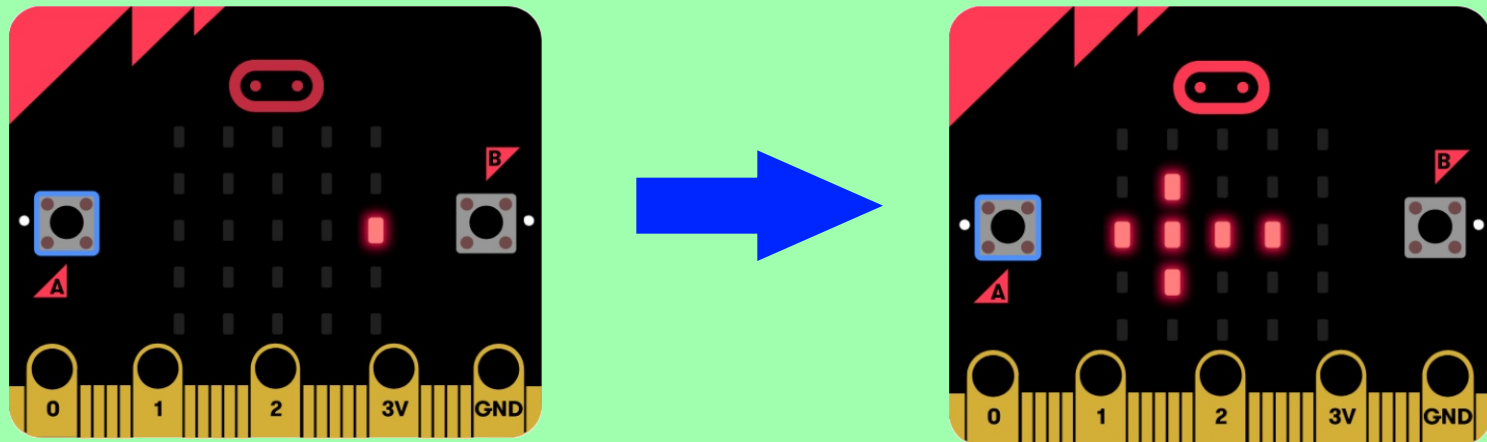
A LED-mátrix azon elemei fognak világítani, amelyeket itt bekapcsolunk.



Különböző égtájak felé mutató nyilakat jelenít meg a kijelzőn. Észak, Észak-kelet, Kelet, Dél-kelet, Dél, Dél-nyugat, Nyugat és Észak-nyugat irányokat ismer.

1. Ismétlés

Készíts egy olyan animációt, amelyben a képen látható nyílvevő megjelenik a kijelző jobb oldalán (kezdetben csak a hegye látszik), majd balra mozog, és megáll, amikor eléri a kijelző bal oldalát.



Kezdő állapot

Végző állapot

2. Animáció eltolással

Az előző feladatban elkészített animáció sok munkával jár, hiszen minden egyes fázist külön meg kellett rajzolni.

A feladat hatékonyabban is megoldható a **képek** kategória néhány blokkja segítségével.

Tanári útmutatás alapján készítsd el úgy az animációt, hogy ezeket a lehetőségeket használod!

Mi történik, ha a nagyobb képet használjuk és a baloldalon helyezzük el az eredeti nyilat?

Hogyan lehetne balról jobbra haladó nyilat animálni?

Készítsünk animációt, amelyben először egy üres, majd egy teli szív érkezik jobb oldalról!

Mi történik, ha az eltolás mértékét, illetve az intervallumot változtatjuk?



2. Rajzok/animációk megjelenítése a LED mátrixon, zenével

3. Akvárium

Készíts egy olyan animációt a kép eltolás módszerrel, amelyben egy kis hal (ami lehet akár 1 pont is) jobbról balra úszik és eltűnik, majd megjelenik egy nagyobb hal, ami szintén jobbról balra úszik és eltűnik (mintha kergetné a kicsi halat). Ezután balról jobbra úszik be egy még nagyobb hal, és tűnőn el a jobb oldalon!

Ügyelj arra, hogy a legnagyobb hal is férjen bele az 5x5-ös területbe, különben nem tud kiűzni a képernyőről a korábban bemutatott módszerrel!

4. Zenélő akvárium

A micro:bit nem csak animációkra képes, hanem zenét is képes lejátszani. Amennyiben a készletben van krokodil csipesz, vagy audió kábel és fülhallgató, akkor közvetlenül a micro:bit tudja szolgáltatni a zenét. Ha nincs, akkor a szimulátorban lehet ezt a lehetőséget kipróbálni. Ez utóbbi esetben is használj a laptpra csatlakoztatott fülhallgatót, különben nagy lesz a hangzavar!

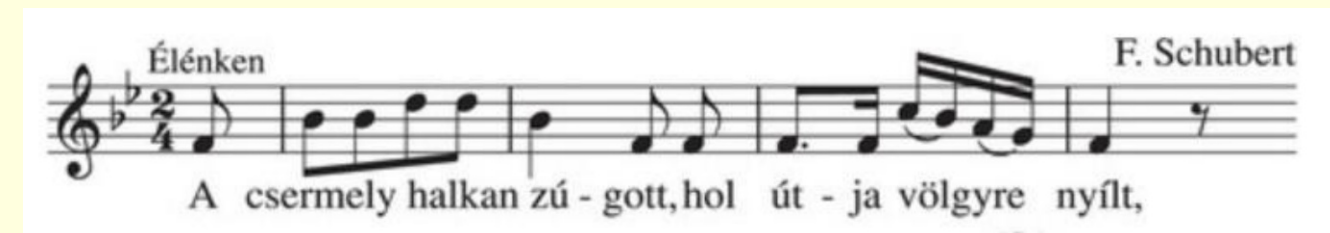
A zene lejátszásához a Zene kategóriában a „start melody” blokkot kell használni. A blokknak két legördülő listája van. Az elsőben az elérhető dallamok közül lehet választani vagy akár saját dallamot is lehet létrehozni, míg a másodiknál négy lehetőség közül választhatunk:

- egyszer történjen meg a dallam lejátszása és utána hallgasson el
- állandóan ismétlődjön a lejátszása a dallamnak
- egyszer játszódjon le a háttérben a dallam
- állandóan ismétlődjön a háttérben a dallam lejátszása

A dallam a háttérben és az előtérben is lejátszható, ezáltal egyszerre több dallam lejátszására is van lehetőség. Amennyiben egy dallamot a háttérben való lejátszásra állítunk, akkor az megszakítható vagy ideiglenesen megállítható. Ha közben elindítunk egy másik dallamot az előtérben, akkor addig a háttérben szóló dallam szünetelni fog, majd ha az előtérben játszott dallam végetért (és nincs végtelenített lejátszása beállítva) akkor a háttérdallam folytatódik.

A dallam lejátszása bármikor megállítható a „stop melody” blokkal, amelynek legördülő listájában be kell állítani, melyik a megállítandó dallam: az előtérben játszott, a háttérben játszott, vagy mind.

Új dallamot is lehet definiálni, amelyhez például az alábbi dallam használható kipróbálásra:



5. Csillaghullás

Milyen hanghatással lehetne aláfesteni egy zuhanó tárgy mozgását? Készíts egy csillaghullás animációt, amelyben a zene/hanghatás kíséri az animációt! Az animációban fentről essenek le pontok (csillagok), mielőtt elérnék az alsó sort, oltódjanak ki.

Az animáció végén jelenjen meg a „Vége” felirat is!

A LED mátrix pontjait egy koordinátarendszer segítségével is kigyújthatjuk, illetve kiolthatjuk. ehhez a **LED** kategória blokkjait kell használnunk.

A LED mátrix minden egyes eleme rendelkezik egy x és egy y koordinátával. Az x érték balról jobbra növekszik 0 – 5 értékkel, míg az y érték fentről lefelé, szintén 0 – 5 értékeket vehet fel. A jobbra levő ábra mutatja az egyes LED-ek koordinátáit.

A koordináták ismeretében az alábbi blokkokkal lehet közvetlenül egy-egy LED-et vezérelni:

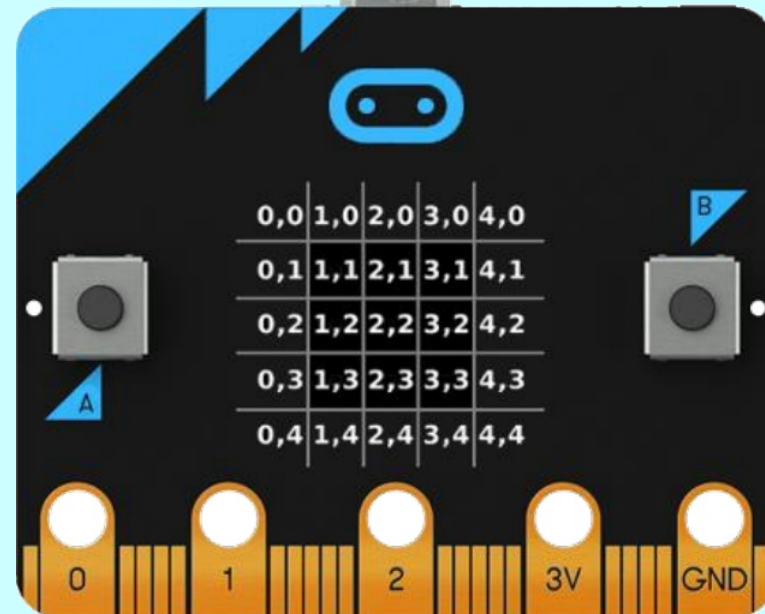
felkapcsol x 0 y 0 A „felkapcsol” (angolul plot) blokk bekapcsolja a megadott koordinátájú LED-et. Hogy pontosan milyen fényerővel fog világítani, azt az éppen beállított fényerő fogja eldönteni.

lekapcsol x 0 y 0 A „lekapcsol” (angolul unplot) éppen az ellenkezőjét csinálja: kikapcsolja a LED-et.

A két állapot közötti váltást teszi lehetővé az „átvált” (angolul toggle) blokk.

A fenti blokkok mindegyike a megfelelő (x,y) koordinátákat várja bemenetként. Ezek természetesen nem csak konkrétan beírt számértékek lehetnek, hanem más blokkok segítségével számítható értékek is lehetnek.

A „felkapcsol” bloknál tud egy kicsit többet a következő blokk, amelynek harmadik



paraméterében megadható, hogy milyen fényerővel világítson az adott LED.

A bekapcsolt LED alapértelmezett fényességét előre is be lehet állítani a „fényerő legyen” (angolul set brightness) blokkal.

Az aktuálisan beállított fényerőt lehet lekérdezni a „fényerő” (brightness) blokkal. Ez nem egy művelet végrehajtását végző blokk, hanem egy számértéket adó blokk, amelynek értékét bárhol, ahova értéket kell írni, fel lehet használni, ha az adott érték helyére ezt a blokkot tesszük.

A fényerő értéke egy egész szám lehet, amely teljesen sötét értéket jelentő 0 és a legvilágosabb értéket jelentő 255 érték között változhat.

6. Toronyház

Van egy 5 szintes toronyház, minden szinten 3 helyiséggel. Szimuláljuk le, hogy a lakók este hazaérnek (véletlenszerű időpontban) és felkapcsolják a villanyt, valamint véletlenszerű időpontban lekapcsolják azt. A ház földszintjén éjjel-nappal nyitva tartó üzletek vannak, így ott mindig ég a villany.

indításkor

felkapcsol x 0 y 4
felkapcsol x 1 y 4
felkapcsol x 2 y 4

Mivel a toronyház alsó szintje mindig ki van világítva, az utolsó sor (y=4) első három oszlopában (x=0, 1, 2) kapcsoljuk fel a lámpákat!

állandóan

átvált x 2 y 2
szünet (ezredmp.) 300
átvált x 1 y 3
szünet (ezredmp.) 300
átvált x 0 y 1
szünet (ezredmp.) 300
átvált x 0 y 3
szünet (ezredmp.) 300

Ezek után néhány pont állapotát változtassuk meg az ellenkezőjére! Erre szolgál az „átvált” (toggle) blokk a LED kategóriában. A váltások között várakozzunk picit a korábban használt „szünet” (pause, Alap kategória) blokk segítségével.

Egy lehetséges megoldás látható a baloldalon.

7. Emeletes ház továbbfejlesztése:

Módosítsd úgy a programot, hogy először a második emelet összes lakásában gyűljön fel a fény balról jobbra, majd a harmadik emelet lakásaiban jobbról balra!

A lámpák kioltása ugyanebben a sorrendben történjen!

8. Véletlen alkalmazása

Milyen jó lenne, ha az ablakokat véletlenszerűen tudnánk kiválasztani! Sokkal életszerűbb lenne az animációnk. Eddig mi határoztuk meg, hogy mikor melyik ablakban gyűljön fény, de ezt most a micro:bitre bízunk.

Ehhez a Matek kategóriára van szükségünk. Abból is a „véletlenszám...” (pick random) blokkra.

Mint a képről látható, a blokk kerek, azaz számértéket ad vissza. Az érték egy véletlenszám, amely a két megadott paraméter közé eső egész szám (beleértve a két határt is, azaz zárt intervallumot használ).

Ezt az értéket bárhova betehetjük, ahol számértéket kell megadni.

Ezzel mind az X, mind az Y értéket véletlenszerűvé tehetjük.

Lent látható a megoldás, amely véletlenszerűen választja ki az ablakot.

Hasonló módon lehet a késleltetést is véletlen értékre állítani...

állandóan
átvált x véletlenszám 0 és között 2 y véletlenszám 0 és között 3
szünet (ezredmp.) 300

8. Fejlesszük tovább a toronyházat!

A Kovács család elutazott, ezért az ő lakásuknak sötétnek kell maradnia. Ők az 1. emelet 3. lakásában laknak. Mit kell tenni, hogy ez az ablak soha ne legyen kivilágítva?

Toljuk el a toronyházat egygel jobbra, hogy középre kerüljön!
Hogyan lehet ezt megoldani feltételvizsgálat nélkül.

A városokban néha előfordul teljes áramszünet is (pl. vihar miatt, vagy mert Godzilla átrágja az áramkábelt). Oldjuk meg, hogy az 'A' gomb megnyomásával átmenetileg teljesen el lehessen sötétíteni a LED kijelzőt, és a 'B' gombbal pedig lehessen visszakapcsolni.

Ehhez a LED kategóriából még egy blokkra lesz szükség:

Ez a blokk (led enable) a LED kategórián belül a „továbbiak” részben található. Attól függően, hogy igaz (true) vagy hamis (false) a paramétere, letiltja vagy engedélyezi a LED-mátrixot. Amennyiben a mátrix le van tiltva, egyetlen LED sem világít rajta. Engedélyezés esetén viszont minden LED az előző fényességével fog újra világítani.

LED engedélyezve hamis

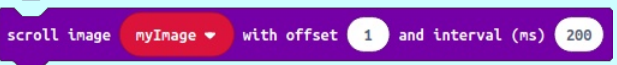


A kép forrása: <https://fmc.hu/2019/06/02/kiraly-egy-fraszt-maradt-volna-a-viz-alatt-godzilla-ii-a-szornyek-kiralya-kritika/>

9. 5x5-ös emeletes ház

Módosítsd úgy a programot, hogy a teljes kijelzőt foglalja el az emeletes ház, vagyis egy 5 emeletes házat valósíts meg, ahol minden emeleten 5 lakás van. A földszinten a bal oldali 3 ablak állandóan világítson, a másik kettő véletlenszerűen legyen be-, illetve kikapcsolva!

A megismert új blokkok



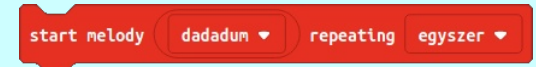
Begörgeti jobbról balra haladva a megadott képet.
myImage: ide kell a begörgetendő képet betenni.
offset: ennyi képpont legyen egy-egy görgetés lépésköze
interval: ennyi ezredmásodperc várakozás teljen el két lépés között



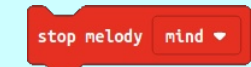
Létrehoz egy képet, amelyet más műveletben (például a görgetésben) fel lehet használni.



Létrehoz egy dupla szélességű képet, amelyet más műveletben (például a görgetésben) fel lehet használni.



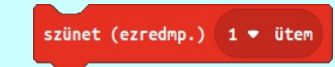
A kiválasztott dallamot (1. legördülő lista) lejátssza a megadott ismétléssel (2. legördülő lista): egyszer az előtérben, állandóan az előtérben, egyszer a háttérben, állandóan a háttérben.



A blokk segítségével meg lehet állítani a lejátszás alatt levő dallamot.
A legördülő lista segítségével választhatunk:
- az előtérben játszott dallam
- a háttérben játszott dallam
- minden dallam leállítása közül.



Az adott hangot a megadott ütemmel lejátssza.



Szünetet tart a megadott ütemig, ami alatt csend lesz.

2. Rajzok/animációk megjelenítése a LED mátrixon, zenével



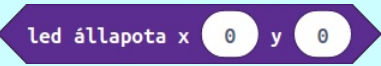
A két paraméterben (x és y) megadott koordinátájú LED felkapcsolása a mátrixon.



A két paraméterben (x és y) megadott koordinátájú LED lekapcsolása a mátrixon.



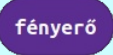
A két paraméterben megadott koordinátájú pont állapotának átváltása. Amennyiben a LED eddig világított, akkor kikapcsolja azt, ha eddig ki volt kapcsolva, akkor felkapcsolja azt.



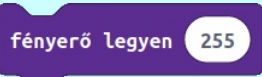
Egy logikai értéket szolgáltató blokk, amelyet tehát oda lehet tenni, ahol logikai érték helyezhető el.
A blokk igaz értéket ad vissza, ha a paramétereivel megadott koordinátájú LED világít, hamis értéket, amennyiben nem világít.



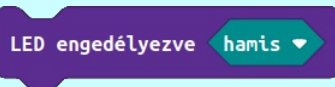
A sima felkapcsolástól csak abban tér el, hogy beállítható a harmadik paraméterben a fényerő is.



Az aktuális fényerősség értékét adja meg. Ezt a blokkot ott lehet elhelyezni, ahol értéket kell megadni. Így bármikor az aktuális fényerősség értékét lehet egy blokkban értékként felhasználni.



A fényerősség beállítása a LED mátrixon. Értéke 0 – 255 változhat. (0: teljesen sötét; 255: a legvilágosabb)

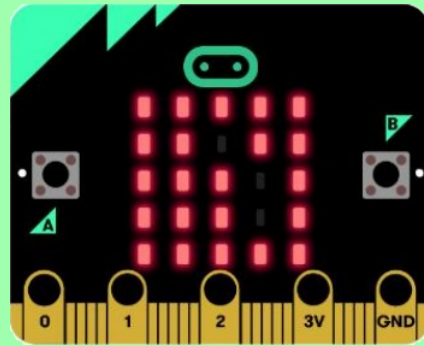


A LED kijelző engedélyezése (true értékű paraméter) vagy tiltása (false értékű paraméter esetén).



A két paraméter által megadott zárt intervallumba eső egész számot állít elő véletlenszerűen. Az így kapott számértéket bárhol fel lehet használni, ahova számértéket lehet írni.
A blokk a Matek kategóriában található.

1. Varázsdoboz



Készíts egy varázsdobozt.

A LED kijelző első és utolsó oszlopa, valamint első és utolsó sora legyen folyamatosan kigyújtva, ez lesz a doboz.

A doboz belsejében lévő 9 pont viszont véletlenszerűen gyúljon ki, illetve oltsódjon ki – mondjuk 500 alkalommal.

A varázsdobozban lévő mozgás csak akkor látszódjon, ha alulról nézzük a micro:bit-et, ha felülről nézzük, akkor csak a doboz látszódjon és ne animálódjon. Az ‚A‘ gombbal lehessen elfektetíteni a kijelzőt, a ‚B‘ gombbal pedig visszakapcsolni azt.

2. Világosság módosítása

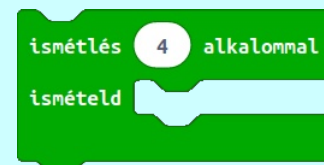
Mint az előző lecke szövegében már volt rá utalás, a micro:bit kijelzőjén megjelenő ábrák világosságértéke is beállítható. Ehhez a LED kategória „fényerő legyen...” (set brightness) blokkját kell használni. Az érték 0 és 255 között vehet fel egész értéket. A 0 a teljes sötétséget, a 255 a teljes világosságot jelenti.

Az éppen beállított fényességértéket le is kérdezhetjük a „fényesség” (brightness) érték-blokkal, amelyet bárhova, ahol számértéket kell megadni, be lehet tenni, így a fényesség értéke felhasználható.

Egyedi LED-ek fényességét annak kigyújtásakor is be lehet állítani a „felkapcsol” blokk három paraméteres változatával, amelynek harmadik paramétere a beállítandó fényerőt várja.

A fentiek felhasználásával készítsünk egy animációt, amely az ‚A‘ gomb megnyomására egy szívdobogást mutat be úgy, hogy a szív alakját változó fényességgel jelzi. A ‚B‘ gomb megnyomására egy átlós szakasz jelenjen meg folyamatosan erősödő világosságértékkel.

A szívdobogás úgy látványosabb, ha nem csak egyszer történik meg. Erre alkalmas a műveletek ismétlése, amelyet a Ciklusok kategória használatával lehet megvalósítani. Most válasszuk az itt található eszközök közül azt, amivel adott alkalommal lehet valamit megismételni és a szívdobogást a kijelző letörlése után négyszer végezzük el!



Ciklusok



A programozás egy nagyon fontos eszköze a ciklus. A vezérlési szerkezetek egyik csoportját alkotja. Ennek a segítségével ugyanazt a műveletet többször is végre lehet hajtani.

A művelet vagy műveletsor, amelyet ismételni kell, a **ciklusmag** nevet kapta.

Többféle ciklus létezik, amelyek közül egyet már korábban is használtunk: a „folyamatos” blokk valójában a bele tett műveleteket a végtelenségig ismétli a háttérben. Ez valójában egy végtelen ciklus.

A ciklusoknál azonban általában valamit meg szokás adni, ami meghatározza, a ciklusnak meddig kell ismétlődnie. Ezt szokás a ciklus feltételének nevezni.

A micro:bit is képes többféle ciklust használni, így például a baloldali példában már említett ciklust, amelynél előre eldönthető, mennyi alkalommal hajtódjon végre a ciklusmag. Ebben az esetben ciklusfeltételnek tekinthető, hogy a ciklusmag végrehajtódott-e az előírt alkalommal.

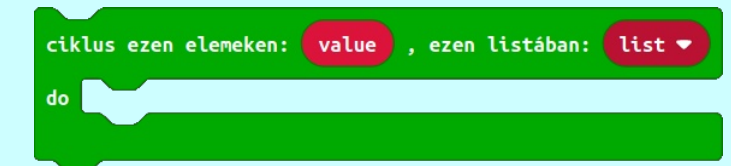
Más ciklusoknál nem lehet előre eldönteni, hányszor fog végrehajtódni, mert a ciklus végrehajtása során dőlhet el, hogy teljesül-e a feltétel – amely lehet kilépést meghatározó vagy a ciklusban maradáshatározó feltétel is.

Amennyiben egy fel-

tétel teljesüléséig akarunk valamit ismételni, akkor arra az „amíg... ismételd” (while... do) ciklus alkalmas. Az „igaz” helyére egy logikai értéket kell megadni.



Az „ismétlés...” blokkhoz kicsit hasonló, de többet nyújtó ciklust lehet létrehozni a „ciklus... 0-tól ...-ig” (for ... from 0 ... to) blokkal, amelyben egy ciklusszámláló is rendelkezésre áll, így bár ezúttal is előre eldöntött, hányszor hajtódik végre, de a 0-tól növekvő számlálónak köszönhetően a ciklusmagban ki lehet deríteni, hanyadik alkalommal fut éppen le.



Kicsit ehhez hasonló, de nem egy számlálót használó ciklus a negyedik blokk által létrehozható. Ebben az esetben a számláló egy értéklista elemein hajtódik végre. A későbbiekben még fogsz ezekkel a ciklusokkal a gyakorlatban találkozni.

További vezérlési szerkezet, amit már ismersz, az **eseményvezérlés**. Ezt használjuk például, amikor a gombnyomásra mint eseményre adjuk meg, mit kell tenni. A későbbiekben még további vezérlési szerkezetként megismered majd az **elágazásokat** is.

3. Animáció számlálós ciklussal

Az előző projektben úgy készítettünk animációt, hogy a fényerő beállítással játszottunk. Most csináljuk meg ugyanezt úgy, hogy az összes fényerősségszintet kipróbáljuk 0 és 255 között!

Ennyi fényerősség szintet kézzel már nagyon idegőrlő lenne beállítani, ezért ciklust kell hozzá használni. De melyiket?

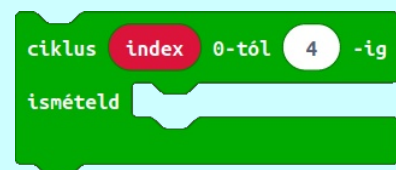
A feladat tipikus példája a számláló ciklusnak (amelyet a legtöbb programozási nyelven „for ciklus” formájában lehet megtalálni). Ezért most egy kicsit részletesebben megismerkedünk ezzel a ciklussal.

A ciklushoz tartozó blokk létrehoz egy számlálót. Ez valójában egy változó, amely nevét onnan kapta, hogy az érték, amelyet képvisel, változhat – ez esetben rendszeresen változik is.

A ciklus belsejében (a ciklusmagban) a változó értéket fel lehet használni. Ebben különbözik az eddig használt ciklustól, ahol csak azt lehetett megadni, hány-szor fusson le, de a ciklusmagban nem lehetett tudni, hol tart a ciklus. Most viszont erre az információra szükségünk van. Amennyiben a piros paramétert nem változtatjuk meg, akkor a ciklusváltozó neve „index” lesz.

Helyben nem is tudjuk megváltoztatni, de a Változók kategória segítségével lehetne új változót létrehozni. Erre a kategóriára szükség is lesz, mivel itt lehet megtalálni azt az érték-blokkot, amely a változónk nevét viseli. Ahol a változó értékét fel akarjuk használni, oda ezt a blokkot kell behúzni.

A változó értéke 0-tól indul mindenképpen és addig marad a végrehajtás a ciklusnál, amíg el nem éri a második paraméterben szereplő egész számot (az ábrán ez négy, de nekünk ide 255 kell).



Hogyan tudnánk megoldani, hogy visszafele (255-től 0-ig) állítsuk be a világosságértékeket?

A Matek kategória blokkjait érdemes megnézni. Itt elég sokféle számítási művelet megvalósítására alkalmas blokkot találunk.

Az egyes számítási műveletek blokkjai mind-mind új számértéket állítanak elő, ezért ezek a blokkok egymásba is ágyazhatóak, amivel bonyolultabb számítások is elvégezhetőek.

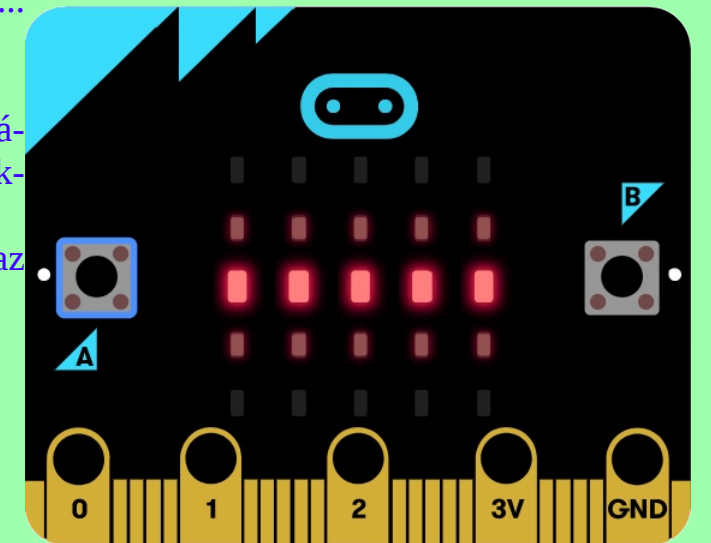
3. Rajzok / animációk a LED mátrixon

4. Különböző világosságú csíkok

Az imént ciklussal változtattuk az ábra világosságértékét. A ciklust viszont arra is használni lehetne, hogy a LED kijelzőn egy sorban az összes pontot egymás után kigyújtsuk. Ehhez a már korábban bemutatott LED kategóriabeli „kigyújt... fénnyességgel” blokkot kell használni.

Készítsd el azt a projektet, amelyben a kijelző középső sora 255-ös világosságértékű pontokból áll, az alatta és a felette levő sorok 50-es értékűekből.

A pontok balról jobbra, egymás után, kis késleltetéssel gyűlnek ki az 'A' gomb megnyomásakor.



5. Animáció több eszközön

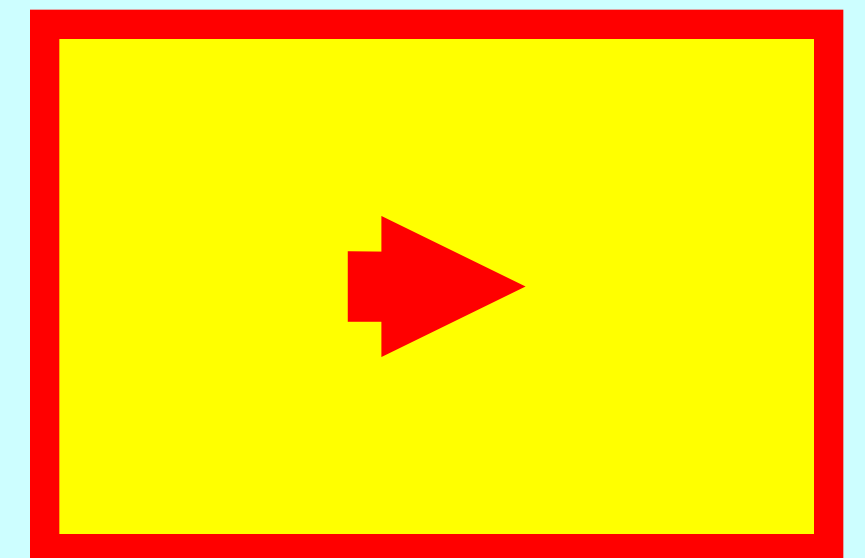
A címsorra kattintva egy Youtube videót nézhetsz meg. A videó hossza 2:27. A videóban több okostelefonon látható egy animáció lejátszása.

Ahhoz, hogy ilyen animációt lehessen készíteni, az eszközöket tökéletesen össze kellett hangolni. Ennek több módja is lehetséges. Az egyik, ha mindegyik eszköz a saját animációját játsza, amelyek tartalma van megfelelően időzítve. A másik megoldás, ha a telefonok kommunikálnak egymással és indítójeleket küldenek egymásnak.

Előbbi esetben nem csak jól kell időzíteni az egyes videókon lejátszódó animációt, de fontos, hogy azok teljesen egyszerre induljanak el.

A második esetben a kommunikációt kell megoldani a telefonok között, és fontos, hogy az indítójelre minden eszköz a megfelelő sebességgel reagáljon.

Micro:bitekkel ilyen kidolgozottságú animációkat, mint a videón látható, természetesen nem tudunk készíteni, de azért nem kell az eszköz lehetőségeit lebecsülni.



5. Animáció több eszközön

Alakítsatok 3-4 fős csoportokat!

A csoportok feladata az lesz, hogy készítsenek olyan animációt, amelyhez 3-4 micro:bit képességeit használják fel. Kezdtetek ötletelni, készítsétek el az animáció forgatókönyvét, hogy a következő alkalommal már csak a megvalósítással kelljen foglalkozni.

Ügyeljetek arra, hogy nem szabad túl bonyolult projektben gondolkodni, hogy a következő alkalom-

mal rendelkezésre álló idő alatt meg lehessen valósítani!

Mivel már ismeritek a micro:bit zenei képességeinek egy részét, érdemes lenne az animációkat zenével is kísélni. Ebben az esetben azonban figyeljetek arra, hogy a zenét a micro:bit csak fülhallgatón keresztül tudja megszólaltatni!

Készíts animációt ciklus felhasználásával, amelyben egy labda (pont) halad a képernyőn vízszintesen balról jobbra, majd a falat elérve visszapattan!

Készíts programot, amely folyamatosan kiválaszt véletlenszerűen egy koordinátát és oda egy véletlenszerűen meghatározott (0 és 50 közti) fényerejű pontot helyez el! Állíts be egy kis késleltetést, hogy a változások jól láthatóak legyenek!

Próbáld leszolmizálni egyik kedvenc dalodat vagy keress rá a kottájára az interneten! Az „A” gomb megnyomásakor játszódjon le a dallam, a „B” gomb megnyomásakor legyen kiírva szövegesen a szerző vagy az előadó neve!

Az újonnan megismert blokkok

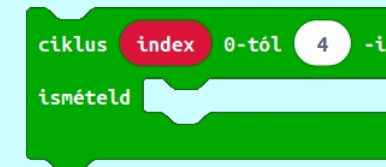


A blokk tartalmát pontosan a megadott számú alkalommal ismétli meg.



Elöltesztelő ciklust valósít meg a blokk:

Addig ismétli a blokk tartalmát, amíg a megadott feltétel igaz.



Számlálós ciklus:

A ciklusváltozó 0-tól a megadott számig veszi fel az értékeket. Mivel nullától indul a számolás, az ismétlés egyel többször történik meg, mint a megadott szám.

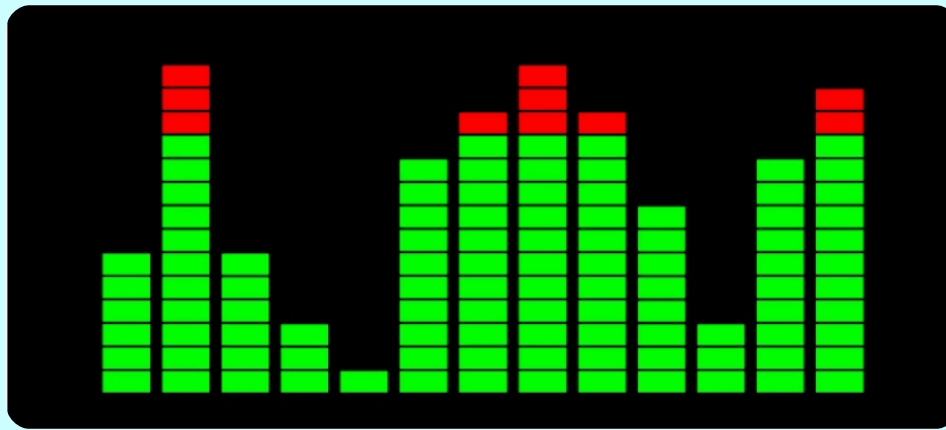


Bejáró ciklus:

A ciklus a megadott lista elemein megy végig.

4. Egymásba ágyazott ciklusok, elágazások használata, változók használata

Készíts programot, amely elhelyez 10 pontot véletlenszerűen kiválasztva a koordinátán, majd letöröl a kijelzőről 20 véletlenszerűen kiválasztott pontot és ezt a két műveletet folyamatosan ismétli!



Hi-Fi eszközökön, zenelejátszó alkalmazásokon gyakran látunk olyan kijelzőt, amely a hang erősségét jelzi ki különböző tartományokban. Készítsünk olyan alkalmazást, amely ezt szimulálja!

Az alul látható képre kattintva elérhető a videó, amelynek nyitóképet mutatja. Ezen a videón látható, hogyan is néz ki egy ilyen jelszintjelző.

Az egyszerűség kedvéért csak szimulálni fogjuk a jelszintek jelzését: a kijelző oszlopaiban véletlenszerűen meghatározott, eltérő magasságú pontsorok raj-

zolásával fogjuk ezt megoldani.

Ahhoz, hogy ez működjön, használnunk kell a korábban tanultakat (rajzolás a LED kijelzőre, ciklusok, véletlenszámok), de szükség van új ismeretekre is, például arra, hogyan hozunk létre változót és adjunk értéket neki.

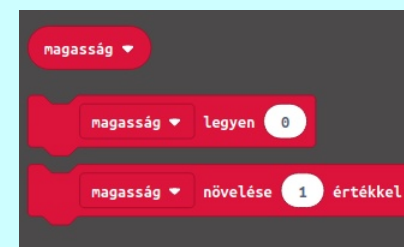
Változó létrehozásához a Változók kategóriába kell menni. Egy üres projekt esetén ezen a részen nem látunk semmi mást, csak egy

gombot a „Változó létrehozása” vagy angol változatban a „Make a variable” szöveggel. Erre a gombra kattintva tudunk új változót létrehozni.

Hozzuk létre a *magasság* nevű változót! A változó létrehozását követően három új blokk jelenik meg a Változók kategóriában. Amennyiben újabb változót hozunk létre, akkor sem fog a blokkok szá-

ma növekedni, ugyanis a változóval végezhető három műveletet jelzi ez a három blokk és mindháromban a listából lehet kiválasztani a használni kívánt változót. A lista minden létrehozott változót mutatni fog.

Az első blokk egy érték-



A változók olyan elemei a programnak, amelyek értéke megváltozhat. Matematikából szintén használják ezt a kifejezést ugyanebben az értelemben. A kettő között a különbség mindösszesen a változók megjelenési formájában van: Matematikában egy változó mindig egy – általában írottak, de legalábbis dőltnek – tűnő betűvel jelölik, mint például ezek: x , y , a , m . Bizonyos változókat akár matematikából, akár fizikából például jelölhetnek görög betűkkel is.

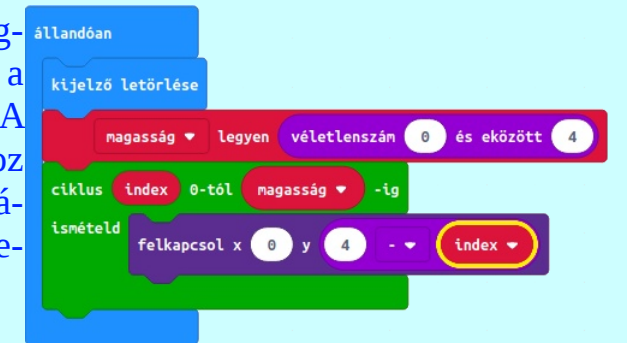
A programokban használt változók általában hosszabb, beszédesebb nevekkkel rendelkeznek, így nem csak egy változtatható értéket hordoznak, de az elnevezésük a funkciójukra, jelentésükre is utal. Ezért fontos, hogy a változók neve mindig tükrözze a funkciójukat!

változó értékét bármely blokkban egy érték megadási helyén felhasználhatjuk. Valójában ez a blokk lekérdezi a változó pillanatnyi értékét.

A másik két blokk segítségével lehet a változó értékét megváltoztatni. Közülük a felső segítségével a pillanatnyi értékétől függetlenül lehet új értéket adni neki (ezt kell a nulla helyére írni), míg az alsó a pillanatnyi értéket növeli meg annyival, amennyi értékként szerepel. Nyilván ez utóbbi értéket negatív értékre állítva csökkenteni lehet a változó értékét.

Az oszlopokat ciklussal fogjuk kirajzolni, viszont mivel a legfelső sor a 0-s indexű, kénytelenek leszünk megfordítani a skálánkat, vagyis a generált véletlenszámot ki kell vonnunk 4-ből ahhoz, hogy a megfelelő eredményt kapjuk (vagyis ha a kapott véletlenszám 4, akkor nem a legalsó sorban (4-es y koordinátára) fogunk rajzolni, hanem a legfelsőn, aminek az y koordinátája 0. Erre az új változóra szükség is lesz, hiszen ez alapján tudjuk kiszámolni, hogy hol kell kigyulladnia a LED-nek az adott oszlopban.

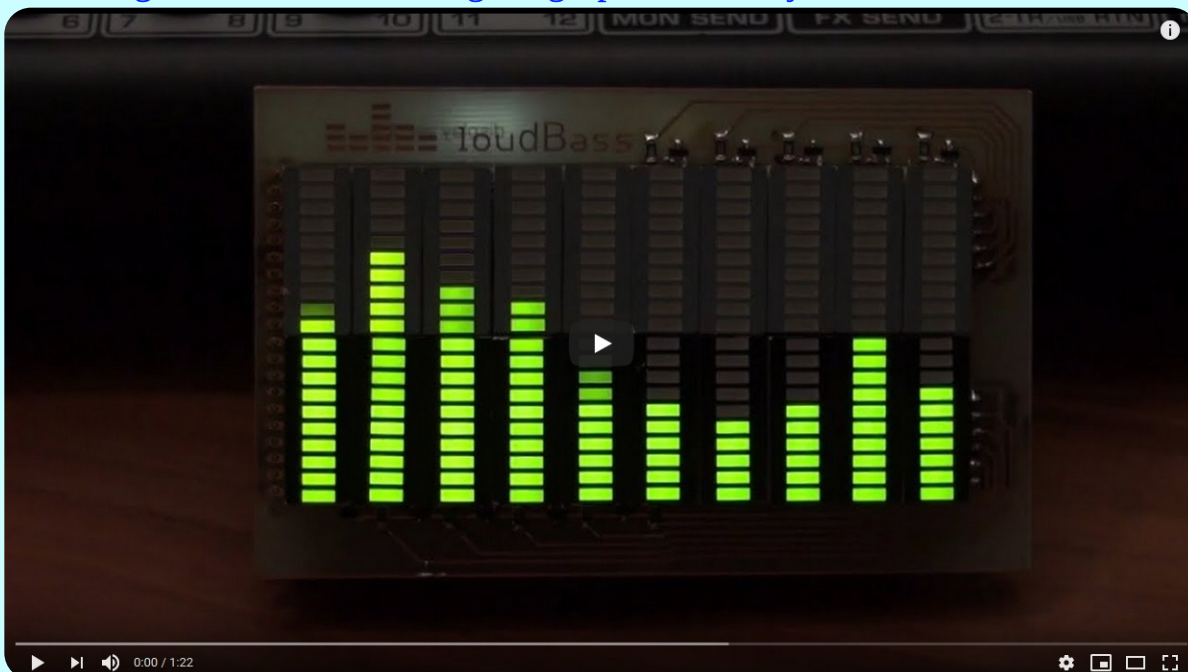
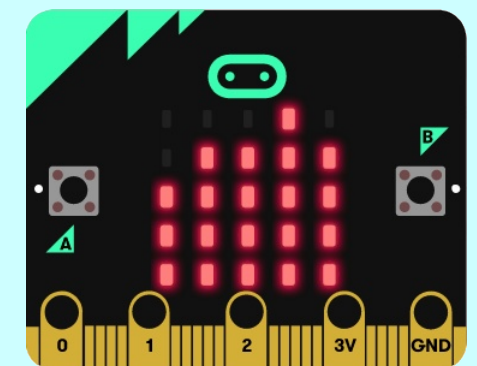
Egy oszlop megjelenítését mutatja a jobboldali ábra. A további oszlopokhoz ugyanezt kell lemásolni az oszlop indexének módosításával.



Figyeljük meg, hogy a számlálós ciklus is tartalmaz egy változót, így amikor ezt a blokkot a programba húzzuk, a változóknál megjelent egy új változó értékblokkja, amely változónak a neve megegyezik a számlálós ciklus számlálójával!

A feladatot kiegészíthetjük zene lejátszásával is, mintha valóban zenére működne az alkalmazásunk.

Bár ez az alkalmazás csak szimulálja a jelszinteket, megfelelő programmal, külső szenzornak az eszközre kapcsolásával akár valódi jelszintmérőt is lehetne a micro:bit-ből készíteni, amely tényleges hangokra reagál...



4. Egymásba ágyazott ciklusok, elágazások használata, változók használata

Alkalmazásunk szépen működik, de a kódja nem elég hatékony. Négyyszer szerepel benne ugyanaz a művelet – igaz, mindegyik másik oszlopra vonatkozik.

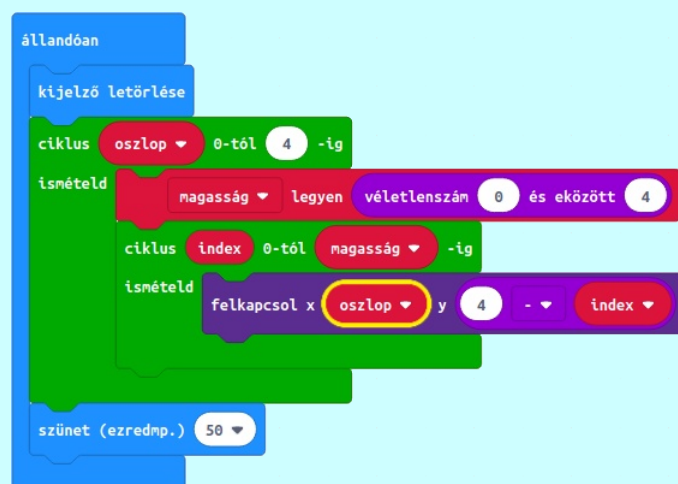
Az ilyen megoldás nem csak azért nem jó, mert sokat kellett hozzá másolni és már-már áttekinthetetlenül hosszú lett, hanem a módosítása is nehéz. Amennyiben a kipróbálása során derül ki, hogy valamit elrontottunk, akkor minden másolatot ki kell javítani. Az egyik alapszabály a programozásban, hogy ugyanazt a kódot (jelen esetben blokkosorozatot) nem írjuk le többször, hanem inkább a programot végrehajtó eszközt (számítógépet, illetve jelen esetben mikrovezérlőt) vesszük rá arra, hogy ugyanazt a kódrészt többször is végrehajtsa.

Igen, erre használatosak a ciklusok! Tehát: itt is valahogy ciklussal kellene a kódrészletek ismétlődését megoldani?

Vegyük észre, hogy a kódunkban az egyes rajzolási blokkok alig különböznek egymástól! A különbség csupán annyi, hogy melyik oszlop sorszáma szerepel a rajzoló blokkban. Ráadásul sorban haladva. Ez már önmagában egy számlálós ciklusért kiált.

De álljunk meg egy pillanatra! Már használtunk ciklust. Most tegyük a ciklust egy másik ciklusba? Miért ne!

Először hozzunk létre egy új változót, *oszlop* néven! Ez fogja tartalmazni, hogy éppen milyen koordinátájú oszlopban kell rajzolni.



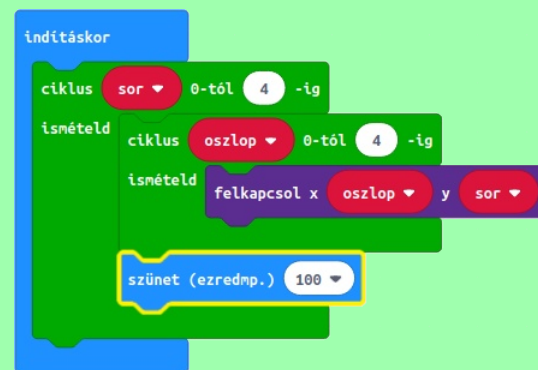
1. Készíts olyan alkalmazást egymásba ágyazott ciklus használatával, amely a LED kijelző összes pontját kigyújtja soronként, balról jobbra haladva! A kigyulladások között legyen annyi szünet, hogy látható legyen a kigyulladás iránya!

2. Módosítsd az alkalmazást úgy, hogy az 1. oszlopban lévő pontok fényereje legyen 1, a másodikban lévő 11, a harmadikban 21, és így tovább! Hogyan lehet a ciklusváltozó értékétől függővé tenni a világosságértéket?

3. Ha ezzel készen vagy, fejleszd tovább az alkalmazást, hogy a fenti rajzolás a ,B' gomb hatására történjen! Az ,A' gomb megnyomásakor pont ellentétes legyen az ábra, vagyis az 1. oszlop legyen a legfényesebb (255) és fokozatosan csökkenjen a fényerő!

4. Az előző alkalmazást úgy módosítsd, hogy az ,A' gomb megnyomásakor a képpontok jobbról balra legyenek bekapcsolva!

Készítsd el az alábbi programot! A következőkben ezt a programot kell majd módosítani.



Mielőtt további feladatokat oldanánk meg, ismerkedjünk meg egy újabb vezérlési szerkezettel, az **elágazás**sal!

Az elágazás lehetővé teszi, hogy valamely feltétel teljesüléséről függővé tegyük egy művelet végrehajtását. Ha a feltétel teljesül, egy adott művelet hajtódik végre, míg ha a feltétel nem teljesül, akkor egy másik művelet hajtódik végre. Van arra is lehetőség, hogy a feltétel nem teljesülése esetén újabb feltételvizsgálatra kerüljön sor, így akárhány különböző esetre szétághazhat a végrehajtás.

Bár úgy tűnhet elsőre, hogy a Feltételek kategóriában két elágazó blokk van, az a két blokk valójában ugyanaz. Mindössze az első változatban a hamis esetre nem kell műveletet megadni, a második változatban a hamis esetre is van a műveletnek hely. A második változatnál levő mínusz gombra kattintva viszont ez becsukódik és az első változattá alakul a blokk. Hasonlóan, az első változaton alul levő plusz gombra kattintva az alsó változattá alakul.

De mi történik, ha az alsó változat alján levő plusz gombra kattintunk? Nos, ebben az esetben azt a fent már említett esetet kapjuk, amikor az eredeti feltétel hamis esetre egy újabb feltételt akarunk megadni. Ez helyett tehetnénk a hamis esetre egy újabb elágazásblokkot is, de így egy blokkon belül lehet láncolni a feltételeket.

Ennél a láncolásnál fentről lefelé haladva az első teljesülő feltételhez tartozó művelet fog végrehajtódni, az összes többi művelet nem.

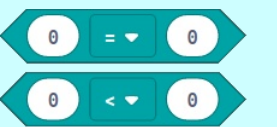
A feltételek láncolása persze nem tesz lehetővé olyan elágazásokat, mint a fentebbi képen látható tábla többszörös elágazásai. Ehhez hasonló esetben tényleg szük-

séges több elágazás-blokk egymásba pakolása.

Szintén a Feltételek kategóriában találhatók további blokkok, amelyek rombusz alakúak. Ezek a kerek szélű blokkokhoz hasonlóan működnek, de míg azok egy számértéket eredményeznek – így ott használhatóak, ahova számot lehet írni –, a rombusz alakú blokkok logikai értéket szolgáltatnak, így oda lehet behúzni őket, ahova igaz (true) vagy hamis (false) értéket lehet tenni.

Két csoportja van ezeknek a logikai blokkoknak:

1. Az első csoport két számérték (vagy két szöveg) összehasonlítására alkalmas. Egyenlő-e a két érték? Valamelyik nagyobb? Ezeket a kérdéseket lehet velük feltenni.



2. A második csoport logikai műveleteket tartalmaz. Ezzel két logikai értéket lehet összekapcsolni. Ha A és B egy-egy logikai érték, akkor például az A és B, A vagy B, nem A logikai műveleteket lehet ezekkel elérni. Ezekről a logikai műveletekről később még fogsz tanulni.



Az elágazás-blokk és az összehasonlító blokkok segítségével el tudjuk érni, hogy az oszlop és sor változók különböző összehasonlításaitól tegyük függővé a LED bekapcsolását.

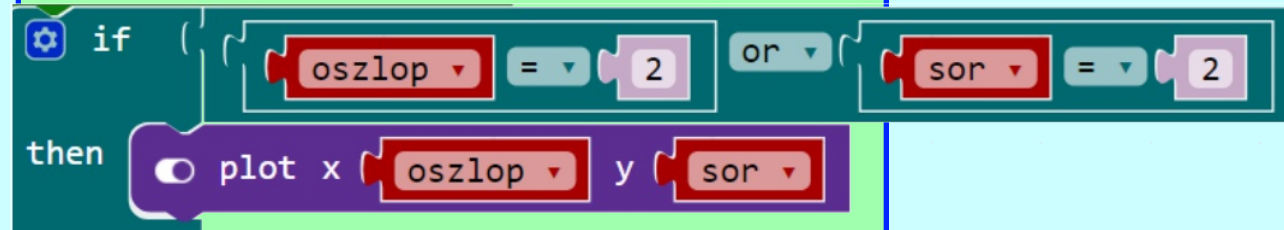
Próbáld ki, mi lesz az eredménye az alábbi összehasonlításoknak:

- oszlop = 2
- oszlop < 2
- oszlop > 2
- oszlop < 2
- oszlop < 2
- oszlop = sor
- oszlop < sor

4. Egymásba ágyazott ciklusok, elágazások használata, változók használata

- oszlop > sor
- oszlop □ sor
- oszlop □ sor
- oszlop □ sor

Ezek után a logikai kifejezéseket is próbáld ki:
oszlop = 2 ÉS sor = 2
oszlop = 2 VAGY sor = 2 (az ábra ezt mutatja)



Miért lesz az az egyes esetekben az eredmény, ami lett?

Milyen feltételeket kell megadni a fenti ábrán látható blokkok módosításával, hogy az alábbi ábrákat kapjunk eredményül? (Csak a legördülő értékeket kell megváltoztatni a fenti blokkban, más módosítás nem szükséges.)

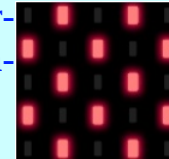
Ügyelj arra, hogy az oszlopok és a sorok 0-tól sorozámozódnak! Próbáld ki, hogy tényleg a megfelelő ábrát kapod-e!

Ha készen vagy ezekkel az ábrákkal, kísérletezz nyugodtan a feltételek megváltoztatásával, és ha kapsz egy érdekes ábrát, jegyezd fel, hogyan sikerült beállítani!

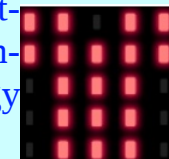
Készítsünk programajánló alkalmazást!

Készítsünk egy alkalmazást, amely programot ajánl nekünk, ha unatkoznánk! Az egyes szabadidős programokból véletlenszerűen válasszon az alkalmazásunk és ikonokkal jelezze, hogy mit kell csinálnunk. Például az alábbi lehetőségeket javasolja nekünk:

Sakktáblát látunk? Sakkozzunk vagy társasjátékozzunk barátainkkal!



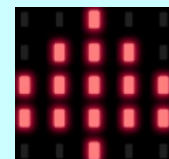
A póló a csapat-sportokra utal. Menjünk focizni vagy röplabdázni!



Ha kottajelet látunk, hallgassunk zenét vagy menjünk koncertre!

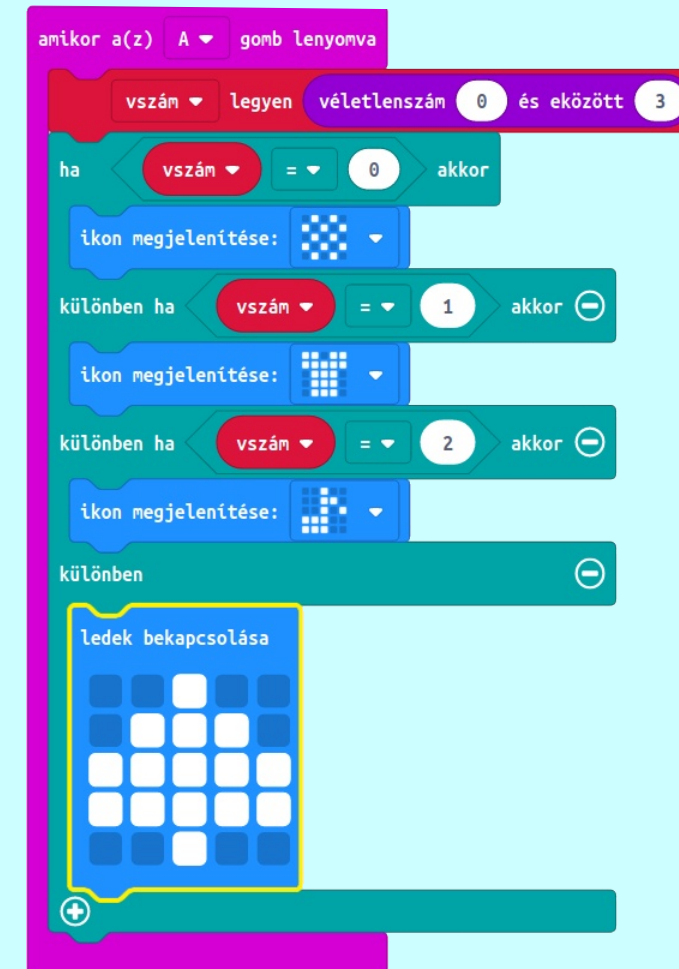
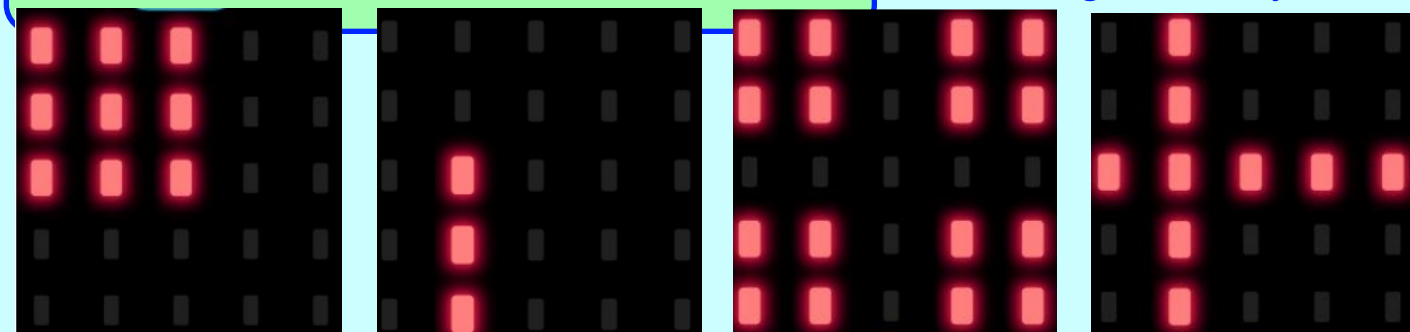


Ha egy fát látunk, menjünk kirándulni!



Generáljunk véletlenszámot akkora intervallumból, ahány programlehetőségünk van, jelen esetben négy (0-tól 3-ig).

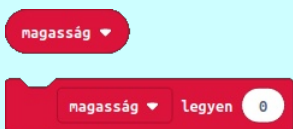
Az elágazás-blokknak most azt a lehetőségét fogjuk kihasználni, amikor több feltételt lehet egymáshoz láncolni, azaz a különben ágból újabb feltételeket nyitunk a plusz jellel. Egészen pontosan egyel kevesebb feltétellel van szükség, mint ahány választási lehetőség van, hiszen amennyiben az utolsó feltétel is hamis, akkor nincs más lehetőség, mint az utolsó lehetőség választása.



Alakítsd át az előző programot úgy, hogy az eszköz megrázásakor véletlenszerűen egy kő, papír vagy olló ikonja jelenjen meg a képernyőn!

Ezt a programot kell továbbfejleszteni úgy, hogy a micro:bit nyilván tudja tartani, hogy hány játékot nyertünk meg. A 'B' gomb megnyomásával növelhessük meg a pontszámunkat, az 'A' gombbal pedig lekérdezhessük az aktuális pontszámunkat. A két gomb együttes lenyomásával a pontszám nullázódjon le.

Újonnan megismert blokkok

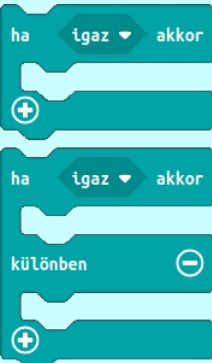


Amennyiben létrehoztunk egy blokkot, akkor ez a három blokk jelenik meg (mindegy mennyi változónk van, ezt a hármat lehet látni).

A legfelsővel lehet egy kiválasztott változó értékét felhasználni ott, ahova egy blokk értéket vár.

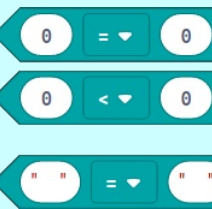
A középső blokkkal lehet az adott változót egy megadott értékre állítani. Az alsó blokkal egy változó értékét növelni (negatív értékkel csökkenteni) lehet a megadott értékkel.

Mindhárom blokkban a használni kívánt változót a listából kell kiválasztani, amelyben minden létező változó neve fel van sorolva.



Az elágazást megvalósító blokk. Bár két külön blokként érhető el, de ezek valójában ugyanazok: a plusz jelre kattintva mindig kaphatunk egy „különben” ágat. Az „igaz” helyén szereplő logikai kifejezés alapján dől el, hogy a művelet vagy műveletek végrehajtására sor kerül-e. Ha a feltétel igaz, akkor végrehajtódik az „akkor” utáni művelet. Ha a feltétel hamis és nincs „különben” ág, akkor a blokk utáni blokkal folytatódik a végrehajtás. Különben ág esetén hamis feltételre az itt szereplő művelet hajtódik végre.

A különben ág végén levő plusz gombra kattintva újabb feltétel adható meg, így a feltételek láncolhatóak egymásba.



Ezek a blokkok két számérték, illetve (a legalsó esetében) két szövegérték összehasonlítására alkalmasak. Mindhárom esetben a középben levő lenyíló listában lehet a szükséges relációt kiválasztani.

Amennyiben a reláció teljesül, akkor igaz (true), ellenkező esetben hamis (false) értéket eredményez a blokk.

A blokkokat minden olyan helyre be lehet tenni, ahova logikai érték szükséges.



Ezek a logikai blokkok is logikai értéket (igaz/hamis) állítanak elő, de a bemenetük (ha van) szintén logikai érték. Az első két blokk valójában azonos, csak éppen nem ugyanaz a logikai művelet van alaphelyzetben beállítva bennük.

Az utolsó kettő blokk szintén megegyezik: ezeknek nincs bemenetük, csupán azt a logikai értéket eredményezik, ami a lenyíló listában beállításra került.

Alul látható a VAGY és az ÉS műveletek igazságtáblája. A NEM műveletnek is el lehet hasonló igazságtáblát készíteni, de az sokkal egyszerűbb: NEM igaz = hamis; NEM hamis = igaz.

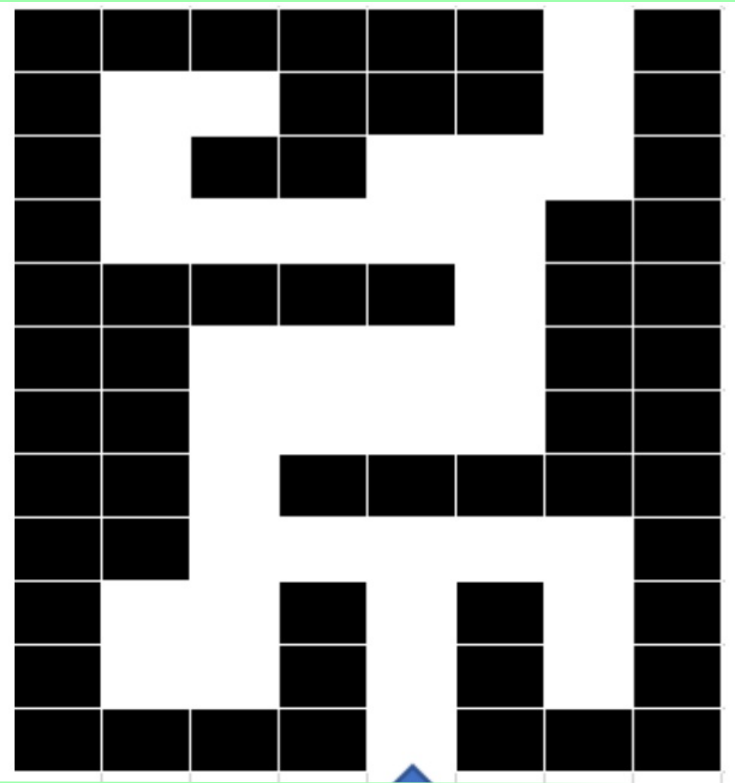
A	B	A vagy B
hamis	hamis	hamis
hamis	igaz	igaz
igaz	hamis	igaz
igaz	igaz	igaz

A	B	A és B
hamis	hamis	hamis
hamis	igaz	hamis
igaz	hamis	hamis
igaz	igaz	igaz

A képen látható labirintusból kell kijutnod, de csak a micro:bit-et használhatod arra, hogy eldöntsd, balra vagy jobbra fordulsz.

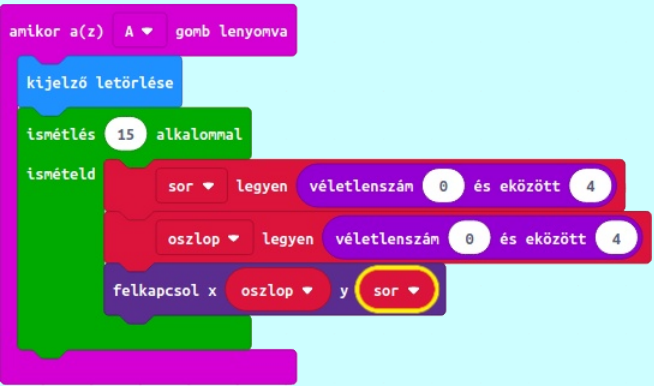
Vagyis készítened kell egy alkalmazást, amely az ,A' gomb megnyomásakor véletlenszerűen egy balra vagy egy jobbra mutató nyilat rajzol a képernyőre. Egy változóban tartsd nyilván, hogy hányszor lett lenyomva az ,A' gomb. A ,B' gomb megnyomásakor legyen kiírva a képernyőre ez a szám. Az ,A+B' gomb együttes megnyomásakor nullázdjon a változó.

A szabály a következő: mindig egyenes vonalban kell haladni, egészen a falig. Ott el kell dönteni, hogy balra vagy jobbra fordulsz és szintén ütközésig kell haladni, ha tudsz. Ha például jobbra tudnál csak fordulni, de a micro:bit azt jelzi, hogy balra kell fordulnod, nem tudsz mit tenni, várnod kell amíg nem jelez egy jobbra fordulat. Ha sem jobbra, sem balra nem lehet tovább menni, akkor a játéknak vége, csapdába kerültél. Ilyenkor kezd újra a játékot a



próbálkozások nullázásával. Ha sikerült kijutni a labirintusból, nyomd meg a ,B' gombot, hogy megtudd az irányváltások számát!

Először induljunk ki az alábbi állapotból. Az ,A' gomb hatására 15 alkalommal véletlen koordinátán pontot gyűjtünk:



Lefuttatva a programot, hány képpont gyullad ki a kijelzőn?
Hogyan garantálható, hogy 15 képpont jelenjen meg?

A LED kategóriában van egy blokk, amellyel le lehet kérdezni, hogy egy adott koordinátájú pont világít-e. Ha igen, akkor igaz (true) értéket, különben hamis (false) értéket ad a blokk.

Ennek a blokknak a segítségével az éppen kiválasztott pontról eldönthető, hogy világít-e már, és ha igen, akkor választhatunk helyette másikat.

Ebben az esetben azonban már nem 15-ször kell a ciklust végrehajtani, hanem addig amíg a 15. pont is kigyújtásra nem került. Erre való a Ciklusok kategória „amíg” blokkja.



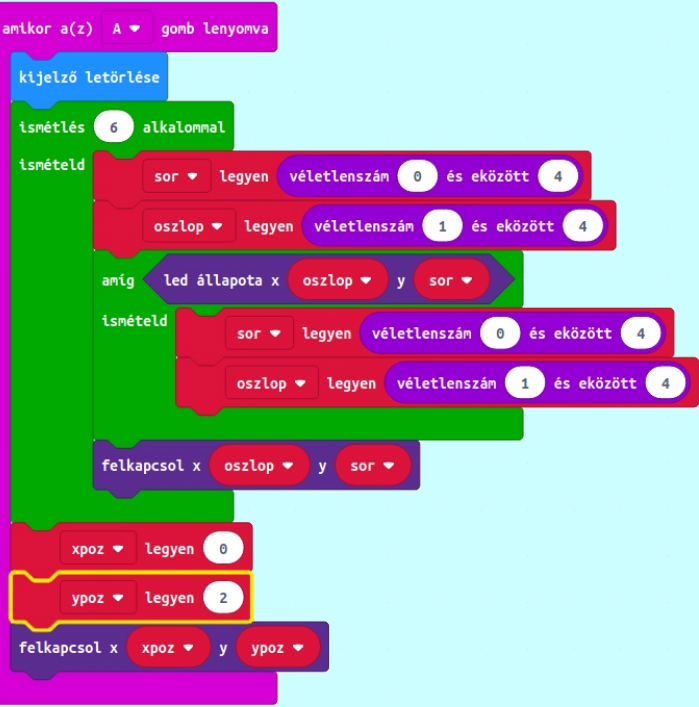
Ha már tudunk labirintust létrehozni, akkor menjünk is végig rajta!

Készítsünk olyan alkalmazást, amelyben végig tudunk vinni egy pontot a labirintuson! Az eszköz négy irányba döntésével fogjuk megváltoztatni a pont pozícióját úgy, hogy csak üres pontra léphetünk.

Kezdetben a pontunk az X=0, Y=2 koordinátán legyen elhelyezve és a jobb alsó sarokba kelljen eljutnunk az akadályok kikerülésével. Hogy nagyobb eséllyel végig tudjunk menni a pályán, 6 pontból álló labirintust generáljunk úgy, hogy az első oszlopba ne tegyünk akadályt.

A jobboldalon látható az alkalmazás első része, amely a labirintust létrehozza.

A program generál egy 6 akadályból álló labirintust, majd az eszköz megdöntésével lehetővé teszi a bejárást. Mindig az X=0, Y=2 koordinátáról kell indulni és a jobb alsó sarokba eljutni.



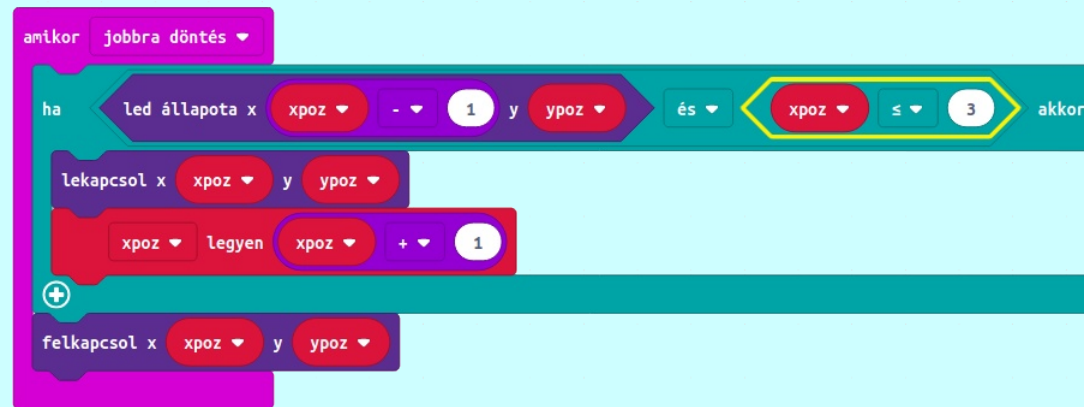
hogy milyen bemenő állításokból az adott logikai művelet milyen eredményt fog adni. Ilyen igazságtáblázatot nem csak az alpműveletekre (NEM, ÉS, VAGY), hanem ezekből összeállított bonyolultabb műveletekre is fel lehet írni.

A legegyszerűbb igazságtáblázata a NEM műveletnek van, hiszen ez egyetlen logikai érték tagadása: ha az eredeti érték igaz (true) volt, akkor hamis (false) értéket eredményez, míg ha az eredeti érték volt hamis, akkor a művelet eredményeként igaz értéket kapunk.

A tagadás műveletét a Feltételek kategóriában, a logikai műveletek között, a „nem” blokk valósítja meg.

A következő oldalon látható a blokk, amely a jobbra döntéshez tartozik.





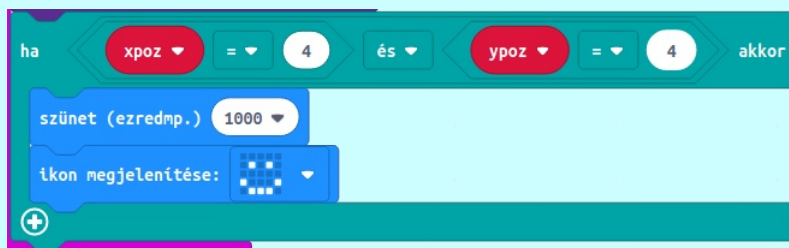
A jobbra döntéskor vérehajtandó műveletek

Bővítsd ki úgy a programot, hogy a pontot az eszköz négy irányba történő döntésével lehessen irányítani!

Ehhez nem kell mást csinálni, mint a jobbra döntés alapján elkészíteni a másik három irányt is...

Jó lenne, ha kapnánk egy szép visszajelzést arról, hogy sikerült a feladatot megoldani, vagyis eljutottunk a jobb alsó sarokba! Ilyenkor 1 másodperc múlva jelenjen meg egy vigyorgó fej!

Ezt megoldhatjuk úgy, hogy az alábbi kódot elhelyezzük mind a 4 blokkunk végén:



Ám ha így oldjuk meg és ki akarjuk cserélni az ikont valami másra, akkor mind a négy helyen módosítani kell a kódot! Ez a megoldás nagyon nem hatékony, az ilyen megoldást pontosan azért kell kerülni, mert több helyen történő módosítás nem csak sok időt igényel, de fennál a veszélye annak is, hogy valamelyik előfordulásban elmarad a módosítás. Ezért azon kódrészletek számára, amelyek többször is előfordulnak, érdemes **függvényeket** létrehozni.

A függvény egy újabb vezérlési szerkezet.

A függvények nagyon hasznosak tudnak lenni amennyiben valamely tevékenységet több helyen is végre kell a prograban hajtani.

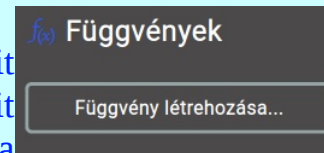
A kategóriák közül a „Haladó” részen belül található a **Függvények** kategória. Ez a kategória a Változók kategóriához nagyon hasonlóan működik: alpból üres és van egy „Függvény létrehozása” feliratú gomb, amivel (meglepő módon) egy új függvényt lehet létrehozni.

Ám a gombra kattintva kicsit másképpen néz ki az ablak, amit kapunk. Ennek oka az, hogy a függvények paraméterrel is elláthatóak. De nekünk paraméterekre most nincs szükségünk. Egyszerűen nevezzük el a függvényünket „ellenőrzés” névre!

A függvény létrehozásakor létrejön a munkaterületen egy blokk (lásd a képen). Ha elsőre nem találod a blokkot, nem kell megijedni: én is kerestem egy darabig, mert elbújt a megjegyzésblokkom alatt. De valahol ott kell lennie a munkaterületen. A függvény nevét egyébként még itt is meg lehet változtatni, így ha elgépelted esetleg, akkor még most gyorsan javítsd ki, mert később, ha már használjuk a függvényt, nem lesz jó ötlet átírni a nevét!

Magában a kategóriában egyetlen blokk jelenik meg: amellyel a függvény meghívását lehet kezdeményezni. Ezt a blokkot húzzuk be a négy eseményvezérlő blokk végére. Vagy másik megoldásként az Alapok kategória „állandóan” blokkjába is behúzhatjuk a függvény meghívását, ebben az esetben a háttérben állandóan ismétlődve fog a függvény kódja végrehajtódni, vagyis folyamatosan újra és újra történik meg a célbaérés ellenőrzése.

hívás ellenőrzés



Nagyon rossz gyakorlat egy programban ugyanazt a kódot többször leírni, hiszen ilyenkor mindig fennál a veszélye, hogy amennyiben szükséges a kódot módosítani (akár mert hibás, akár mert változtatni kell a program működésén), akkor nem minden helyen történik meg megfelelően a kód módosítása.

Ennek a kódismétlésnek az elkerülésére találták ki a függvényeket, amelyek segítségével a kódismétlés elkerülhető, hiszen az adott kódot csak egyszer, a függvény definiálásakor kell megadni és mindenhol ahol a kódot használni kell, csak a függvényt kell meghívni.

Amennyiben nem mindig kell pontosan ugyanazt a kódot használni, a függvények még akkor is használhatóak maradnak, mivel lehet a függvényeket paraméterezni is.

A paraméter egy olyan változó, amelyet a függvényen belül lehet használni. A függvény meghívásakor kell a paraméter értékét megadni az erre rendelkezésre álló konnektorokban: ha szám típusú paramétert definiálunk, akkor számot vár, logikai típusú paraméter esetén csúcsos végű konnektorban várja a logikai értéket.

A függvényt definiáló blokkon belül a paramétereket rendes változóként érhetjük el, de ezek a „változók” a függvény blokkján kívül nem léteznek! A paramétereket éppen ezért nem a Változók kategóriában kell keresni: ott nem szerepelnek. Egyszerűen a függvénydefiniáló blokk elejéről kell a blokkot a megfelelő helyre húzni. Ekkor a paraméter másolatát kapjuk, amit a megfelelő helyre lehet illeszteni.

Alakítsatok párokat és beszéljétek meg, hogyan tudátok továbbfejleszteni a labirintus játékot, hogy a játékelmény jobb legyen! Kezdjétek el az ötlet megvalósítását!

Eddig a következő vezérlési szerkezeteket ismerted meg:

- **Szekvenciális vezérlés:** Ugyan a nevével még nem találkoztál ebben a dokumentumban, de azért, mert ez az alaphelyet, amikor az egyes műveleteket egymás után (szekvenciális sorrendben) kell végrehajtani úgy, ahogy azok fel vannak sorolva.

- **Ciklusok:** Valamely műveletsor ismétlését teszik lehetővé. Az ismétlések száma előre meghatározott lehet, vagy azt, hogy kell-e még ismételni, valamilyen feltétel teljesülésétől tesszük függővé.

- **Elágazás:** Valamely feltétel teljesülése alapján dől el, hogy egy adott műveletsort végre kell-e hajtani. Lehetséges olyan megoldás, amelynél a feltétel csak egy műveletsor elvégzésének szükségességét határozza meg, vagy lehetséges a feltételtől függően más-más műveletsor elvégzését előírni. Utóbbi eset több feltételre az *esetsztérválasztást* jelenti, ahol a feltételek egymás után láncolásával lehet több esetet megkülönböztetni.

- **Eseményvezérlés:** Valamely esemény bekövetkezésétől függ egy adott műveletsor végrehajtása. A micro:bit esetében az események lehetnek valamelyik gomb megnyomása, az eszköz megdöntése, de további események is létezhetnek. Egy weboldalhoz is tartoznak események: amikor az egeret a felhasználó egy adott terület fölé viszi, vagy rákattint az egérrel az oldal valamelyik részére.

Ezekhez jön most hozzá még egy vezérlési szerkezet, a **függvényhívás**. Ezt szokták más néven *eljáráshívásnak* vagy *szubrutinhívásnak* is nevezni. Ebben az esetben a függvény (vagy bizonyos nyelvekben az eljárás) egy műveletsort összefog egyetlen utasítássá, amely utasítást bárhol kiadva (ez maga a függvényhívás) a függvényben definiált műveletsor hajtódik végre. Előnye, hogy így a függvényben megadott kódot csak egy helyen kell a programban szerepeltetni.

Egy függvényt definiál a blokk. A függvény az ebbe a blokkba elhelyezett blokkok által megadott műveletek végrehajtásából fog állni. Létrehozni egy ilyen blokkot a Haladó/Függvények kategóriában a „Függvény létrehozása” gombra kattintva lehet. A létrehozott a függvénynek nem csak nevet, hanem paramétert is lehet adni (nem látszik a blokk ábráján). A létrehozáskor a blokk azonnal megjelenik a munkaterületen, de esetleg más blokkok vagy megjegyzés eltakarhatja azt!

Az alsó ábrán a blokk azon változata látható, amelyen minden lehetséges paraméterből egy-egy szerepel. Az első paraméter szöveges adatnak, a második logikai értéknek, a harmadik számnak, a negyedik a később megismerendő sprite-nak szolgál megadási helyet. Az egyes paraméterek neve látható a piros részben, amit a függvény definiálásakor adunk meg.

Ezzel a blokkal lehet a függvényt meghívni, amelynek neve a blokkban szerepel. Mivel itt a változókkal ellentétben nem szerepel legördülő lista, máris sejthető, hogy ahány függvényt létrehozunk, annyi ilyen blokk lesz a Függvények kategóriában.



<https://hu.pinterest.com/pin/593138213393020610/>



<https://www.wallpaperflare.com/magic-the-gathering-labyrinth-lightning-maze-hd-maze-game-fantasy-wallpaper-mhruj>



<https://www.wallpaperflare.com/theseus-gray-labyrinth-artistic-fantasy-one-person-architecture-wallpaper-pkv0l>

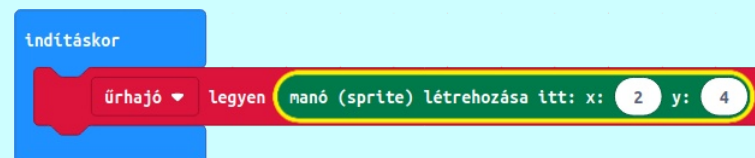
Az elmúlt alkalommal a pont koordinátájának növelésével, csökkentésével készítettünk játékot. Most térjünk rá arra, hogy milyen haladó lehetőségek vannak játékok készítésére!

Készítsünk egy játékot, amelynek a lényege: A kijelző alsó sorában jelenjen meg egy pont. Ez lesz az űrhajó. Az „A” és „B” gombokkal lehessen balra és jobbra mozgatni. Fentről jöjjön meteor véletlenszerű helyről, amelyet ki kell kerülni. Amikor a meteor elérte az alsó sort, kerüljön a felső sorba véletlenszerű helyre. Ütközéskor érjen véget a játék. Számoljuk, hogy hány me-teort sikerült kikerülni. A játék 20 másodpercig tartson.

Először ismerkedjünk meg azzal, hogyan mozgathatunk egy pontot a legegyszerűbben a képernyőn! Ehhez úgynevezett sprite-ot kell használnunk. A sprite egy olyan rajzobjektum, aminek van iránya, tud előre és hátra lépni, lekérdezhetjük, hogy ütközött-e más rajzobjektumokkal, vissza tud pattanni a falról és így tovább. Úgy lehet elképzelni, mint egy LED méretű kis lényecske.

A Haladó/Játék kategória blokkjaival fogunk megismerkedni. Itt találhatóak a sprite-okhoz kapcsolódó blokkok is és még más, játékoknál hasznos blokk is.

Az űrhajó részére hozzunk létre egy sprite-ot, amelynek kezdő koordinátája legyen az X=2, Y=4, azaz alul, középen legyen elhelyezve.



Ezután beállíthatjuk, hogy az „A” gomb megnyomására lépjen hátra egyet az űrhajó (mivel jobbra néz, ezért igazából ennek hatására a kijelzőn balra lép egyet).

Hasonlóan a „B” gomb hatására pedig előre (jobb oldali irányba) kell mennie.



a labirintusnál.

- Folytassuk a játék fejlesztését a következőképpen:
- A játék kezdetekor válasszunk egy véletlenszerűen meghatározott x koordinátát a meteor számára (*meteorx* változó).
 - A meteor számára hozzuk létre a *meteor* nevű sprite-ot.
 - Mivel létrehozásakor a sprite jobbra néz, el kell forgatni 90 fokot, hogy lefele nézzon, hiszen fentről lefelé kell majd esnie.
 - Állítsuk be a kezdeti pontszámot (*score*) nullára.
 - Az űrhajó részére már létezik a sprite, ennek a létrehozása jönne itt.
 - Indítsunk el egy számlálót (*counter*), hogy legfeljebb 20 másodpercig tartson a játék. 20 másodperc elteltével automatikusan megjelenik az elért pontszám.

A fenti felsorolásban leírt művelet sor tulajdonképpen egy tevékenység algoritmus. Későbbi tanulmányaid során az *algoritmus* kifejezéssel még fogsz találkozni. Az elnevezés egy olyan tevékenység leírására használatos, mint amilyen például a fenti is, amely valamilyen eszközzel – most éppen magyar mondatok formájában – leírja egy tevékenység lépéseit általános formában.

Az algoritmus szó Muhammad ibn Musa al-Khwarizmi (kb. 780. – kb. 850.) nevéből származik (egyéb-ként az al-Khwarizmi rész a névben azt jelenti, hogy Khwarizmi városából származik). Ő az a perzsa matematikus volt, akinek köszönhető az indiai eredetű számok használatának európai elterjedése.



A sprite az angol nyelvben az európai fantasy irodalom mitológiai eredetű természetfölötti lényének elnevezése, gyakran egyfajta tündérnek írják le vagy így ábrázolják. Máskor valamifajta éteri lényként szerepel.

Az angol sprite szó a latin spiritus (lélek) szóból ered. A francia nyelvben a szó: *esprit*. A kifejezés több variációja ismert a kelta „spriggen”-t is beleértve. A szó az európai folklórban leginkább a tündérekkel és az elfekkel kapcsolatban fordul elő. A modern angol nyelvben már ritkán használják lélek jelentéssel.

Valószínűleg a szellemszerűség volt az oka, hogy a képernyőn egyfajta szellemként mozgatható képet sprite-nak nevezték el. Például a Commodore számítógépeken volt a sprite egy olyan kis kép, amelynek csak a koordinátáit kellett megadni és a megjelenítéséről maga a számítógép gondoskodott. A géptípusra írt játékprogramokban nagyon elterjedt volt a sprite-ok használata a mozgatandó rajzelemek

megvalósítására.

A micro:bit esetében a sprite csak egyetlen pixel, azaz egyetlen LED pont méretű, de nagyjából ugyanazokat a funkciókat nyújtja, mint ezeknél a 80-as évekbeli számítógépeknél. Valamiért a micro:bit magyartói a sprite-ot nem tündérnek, hanem manónak magyarázták...

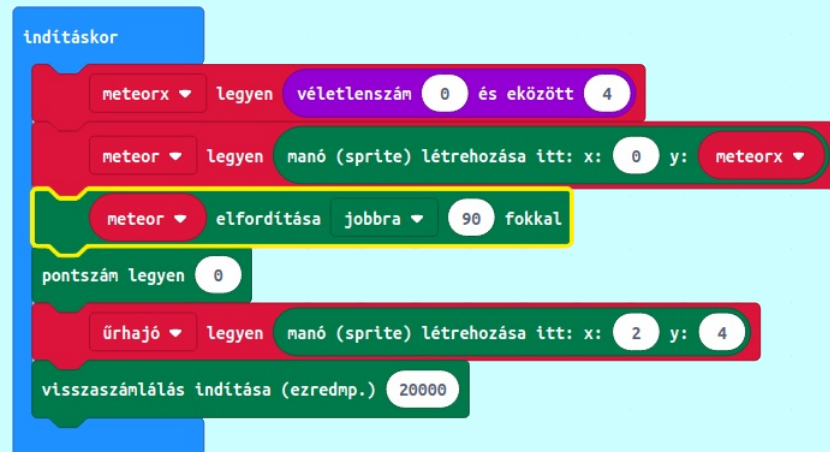


A herceg köszönti a Víz-tündért (sprite-ot) a *Senki Hercegnő*: Egy tündérföldi mese című műben.

(The prince thanking the Water sprite, from *The Princess Nobody: A Tale of Fairyland* (1884) by Andrew Lang)

Az oldalon található képek forrása:

- <https://hu.pinterest.com/pin/454089574903113052/>
- <https://www.pngflow.com/en/free-transparent-png-yyoty>
- https://en.wikipedia.org/wiki/Muhammad_ibn_Musa_al-Khwarizmi



A fenti blokkelrendezést kell kapnunk. Az egyes újonnan előkerült blokkok leírását szokás szerint megtalálod a lecke végén.

Leginkább új elemnek a pontszám számít. Ez egy változóhoz hasonló számláló, amelyet automatikusan hoz létre a micro:bit a játékhoz. Az értékét be lehet állítani, lehet növelni vagy csökkenteni, illetve ki lehet írni.

Az igazi előny a használatában a játék állapottal együtt érződik: ha a játékra beállított idő letelik vagy más okból a játék véget ér, akkor a micro:bit automatikusan kiírja az elért pontszámot – feltéve, hogy az valamilyenkor be lett állítva.

A pontszám beállítását a „pontszám legyen...” blokkal lehet elvégezni. Módosítani a pontszámot a „pontszám növelése... ponttal” blokkal lehet. Amennyiben pozitív értéket adunk meg, akkor növeljük a pontszámot, negatív értékkel pedig csökkentjük.

A pontszámot le is lehet kérdezni, hiszen változóként viselkedik: a „pontszám” blokk adja a pillanatnyi értékét.

Játék számára időkorlát állítható be, ha a játék elindítását a „start countdown” angol nevű, magyarul a „visszaszámlálás indítása...” blokkal végezzük. Az

ebben a blokkban beállított ezredmásodperc letelével a játék

automatikusan véget ér és a micro:bit a „GAME OVER” szöveget, majd az elért pontszámot kiírja. Ugyanezt lehet elérni a „vége a játéknak” blokkal is.

Következő lépésként el kell indítani egy háttérfolyamatot, amely folyamatosan fut. Ugyebár erre használjuk az „állandóan” blokkot.

- Beállítjuk, hogy a meteor lépjen előre egyet.
- Ha az űrhajó hozzáért a meteorhoz, akkor a játéknak véget kell érnie.
- Legyen egy kis késleltetés, hogy ne mozogjon olyan gyorsan a meteor (pl. 300 ms).
- Ha a meteor elérte az utolsó sort (y=4), akkor majd újra meg kell jelennie a felső sorban egy véletlenszerűen választott x koordinátán.
- Amennyiben új meteort jelentetünk meg, akkor egyben a pontszámot is növelni kell egyel.

A blokkok megadása helyett most csupán a még szükséges blokkok ismertetése következik. Ez és a fenti algoritmus alapján a blokkok összeállíthatók.

A sprite valamelyik tulajdonságát le lehet kérdezni ezzel a blokkal. Az első legördülő listában azt a változót kell beállítani, amely a kérdéses sprite-ot tartalmazza. A második legördülő listában pedig a lekérdezni kívánt tulajdonságot lehet beállítani. Ez lehet az x vagy y koordináta, iránya (amerre néz), a fényereje, villogjon-e. A sprite fényerején ugyanazt értjük, mint amikor egy-egy képpont kigyújtásánál beállítjuk a fényerőt. A villogás egy logikai érték: ha igaz (true) a sprite-nak megfelelő képpont villogni fog.

A fenti tulajdonságok bármelyikét be lehet a sprite x értéke legyen 0

Hasonló módon lehet külön azoknak a tulajdonságoknak az értékét növelni egy megadott értékkel, amelyek számértékek (tehát a villogást nem).

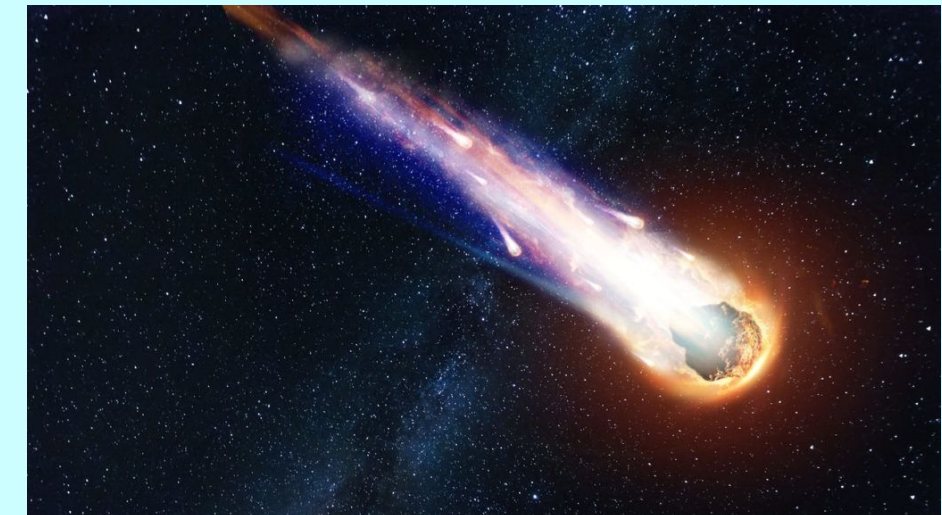
Lehet vizsgálni egy logikai értéket adó blokkal, hogy az adott sprite érint-e egy másikat. Itt tulajdonképpen logikus lehetne két egyforma paraméter szerepeltetése, hiszen mindkét paraméternél sprite-ot tartalmazó változót kell megadni.

Ellenben a másik logikai blokkban csak egy sprite szerepel, hiszen ebben azt lehet vizsgálni, hogy a sprite valamelyik szélén tartózkodik-e a LED-mátrixnak.

Fejleszd tovább a játékot az alábbiak szerint:

- Legyen az űrhajónak pajzsa! Ez azt jelentse, hogy pl. két ütközést kibír az űrhajó és csak a harmadik ütközéskor ér véget a játék!
- Az űrhajót ne csak az ‚A’ és ‚B’ gombbal lehessen irányítani, hanem az eszköz balra, illetve jobbra döntésével is! Mindezt úgy oldd meg, hogy nem használj kódkeverést!
- Adj hangeffektet a projekthez!

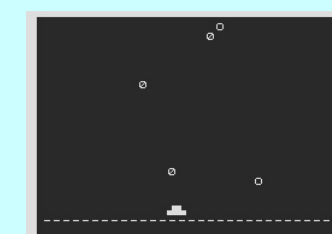
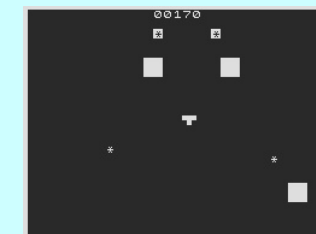
Hogyan lehetne továbbfejleszteni a játékot, hogy növekedjen a játékelmény? Alkossatok párokat! Ötleteltek közösen a játékkal kapcsolatban és valósítsátok meg az elképzeléseiteket!



Azoknak a játékoknak, amelyben a fentről aláhulló meteorokat kell egy űrhajóval kerülgetni, nagy hagyománya van. Az első számítógépes játékok közé sorolható. A legelső személyi számítógépeken ugyan még grafika megjelenítésére nem volt lehetőség, csak a gép karakterkészletét lehetett használni, de így is több ilyen játék készült. A meteor egész egyszerűen egy csillag karakter lehetett, míg az űrhajó egy vagy több egymás

mellé tett aláhúzás-jel vagy kötőjel volt.

Az alábbi három kép egy-egy ZX81 személyi számítógépre készült játékot mutat. A baloldali neve: Meteors II., míg a középsőé: Asteroids. Jobbra szintén ebből a gyűjteményből a Meteor Storm nevű, 1982-ben készült játékot mutatja. Mindhárom kép a <http://www.ep128.hu/ZX81/Games.htm> címről származik, ahonnan ezek a játékok letölthetőek



Újonnan megismert blokkok

manó (sprite) létrehozása itt: x: 2 y: 2

A két konnektorában megadott koordinátákon létrehoz egy új sprite-ot. A sprite-ot bele kell tenni egy változóba, mint értéket. Az újonnan létrejövő sprite jobbra néz.

delete sprite ▼

Törli a sprite-ot, amely a megadott változóban található.

sprite ▼ léptetése 1 lépéssel

A megadott változóban tárolt sprite-ot a megadott lépéssel lépteti abba az irányba, amerre a sprite néz. Negatív érték is megadható, amely esetben a sprite tolatni fog.

sprite ▼ elfordítása jobbra ▼ 45 fokkal

A megadott változóban tárolt sprite-ot elforgatja a megadott irányban a megadott fokkal. A fok értéke nincs korlátozva, azaz bármilyen irányba el lehet forgatni a sprite-ot.

sprite ▼ x ▼

Lekérdezhető a blokkal a kiválasztott sprite valamelyik tulajdonsága. Az első legördülő listában azt a változót kell beállítani, amelyik a kérdéses sprite-ot tartalmazza. A második listából pedig a lekérdezni kívánt tulajdonságot kell beállítani.

A lekérdezhető tulajdonságok: x koordináta, y koordináta, irány, fényerő, villog-e a sprite. Az utolsó tulajdonság logikai érték, a többi szám.

sprite ▼ x ▼ értéke legyen 0

Az első listában kiválasztott változóban tárolt sprite-nak a második listában kiválasztott tulajdonságát lehet a harmadik paraméter értékére beállítani.

sprite ▼ x ▼ növelése 1 értékkel

Az első listában kiválasztott változóban tárolt sprite-nak a második listában beállított tulajdonságát (kivéve a villogást, mert az logikai érték) lehet a harmadik paraméterben szereplő számmal megnövelni. Természetesen ha ez a szám negatív, akkor csökkentés lesz az eredmény.

6. Játékfejlesztés

pontszám legyen 0

Beállítja a pontszámot a megadott értékre.

pontszám növelése 1 ponttal

A pontszámot a megadott értékkel növeli (negatív érték esetén értelemszerűen csökkenni fog).

pontszám

A pontszám értékét adja vissza.

viSSzaszámlálás indítása (ezredmp.) 10000

Elindít egy viSSzaszámlálást a megadott időtartamra. A viSSzaszámlálás végén a játék automatikusan véget ér.

vége a játéknak

A blokk hatására a játék befejeződik.

Tenisz meccs

Készítsünk egy komplexebb játékot lépésről lépésre! Ennek során felhasználjuk a sprite-okban rejlő lehetőségeket, valamint gyakoroljuk a korábban tanultakat.

Egy tenisz meccset fogunk szimulálni. A LED kijelző felső sorából indul a labd, amelyet vissza kell ütnünk a legalsó sorban elhelyezett teniszütővel. A teniszütő balra és jobbra tudjon mozogni. A mozgást az „A” és „B” gombbal, valamint az eszköz balra és jobbra döntésével lehessen elvégezni.

Ha nem sikerült visszaütni a labdát, legyen vége a játéknak. Ha sikerült visszaütni, akkor növekedjen a pontszámunk. Amennyiben a labda visszatér a legelső sorba, véletlenszerűen változtassuk meg a helyzetét, hogy ne legyen túlságosan kiszámítható a játék.

A játék kezdete:

1. A játék kezdetekor állítsuk be a pontszámot nullára.
2. Készítsünk *teniszütő* néven egy sprite-ot és helyezzük el a legalsó sorban, középen.
3. Hozzuk létre a labdát is, a legelső sorban, de véletlenszerűen választott helyen. Mivel a sprite alap esetben jobbra néz, fordítsuk el jobbra 45 fokra, hogy átlósan lefelé induljon majd el.

Az ütő mozgatása:

Gondoskodjunk arról, hogy az ütőt lehessen balra és jobbra mozgatni az „A” és „B” gombokkal, valamint az eszköz balra és jobbra döntésével.

A játék motorja:

A folyamatosan végrehajtandó tevékenységek, amelyek az „állandóan” blokkba helyezendőek, amelyek gondoskodnak a játék folyamatos működéséről, amit a játék motorjának szokás nevezni. A következő tevékenységeket kell itt megvalósítani:

- A labdát előre kell léptetnünk 1 lépéssel.

- Amennyiben a falhoz ért, akkor vissza kell pattannia.

- Ellenőriznünk kell, hogy vége van-e a játéknak. A játék akkor ért véget, ha nem sikerült visszaütnünk a labdát. Ez akkor következik be, ha a labda elérte a legalsó sort ($y=4$), viszont a labda és az ütő x koordinátája nem egyezik meg, vagyis különböző helyen van a labda és az ütő. Egyelőre csak egy függvényhívást helyezünk el itt „vége a játéknak?” néven, később fogjuk a tartalmát megfogalmazni.

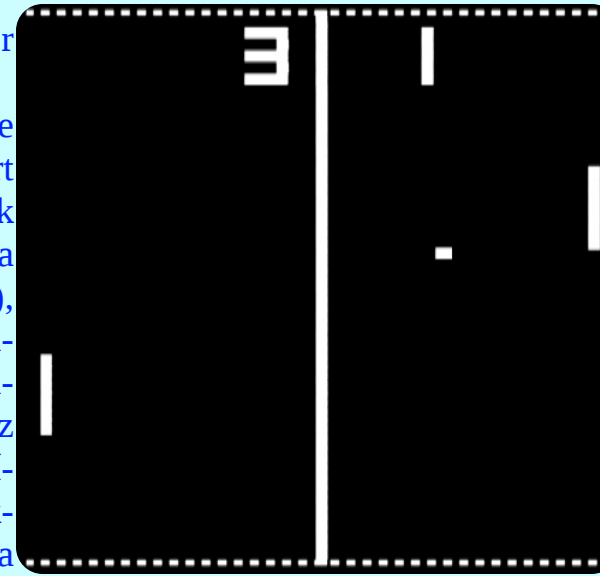
- Szintén ellenőriznünk kell, hogy sikerült-e visszaütni a labdát. Ez akkor következik be, ha az ütő érinti a labdát. Ekkor a pontszámot meg kell egyel növelni. Ezt az ellenőrzést írjuk majd a „sikerült visszaütni?” nevű függvényben.

- Gondoskodnunk kell arról is, hogy amennyiben a teniszlabda visszajutott a legfelső sorba, akkor egy új, véletlenszerűen választott helyről induljon útjára. Ez azért szükséges, hogy a játék élvezetesebb legyen. Ennek hiányában könnyen előfordulhatna, hogy a labda újra és újra ugyanazt az útvonalat járja be. Ezt a kódot fogjuk az „új teniszlabda” függvénybe írni.

- A blokkunk végén érdemes elhelyezni egy kis késleltetést, különben túl gyors lenne a játék.

A fentiek megvalósításához szükséges valamennyi ismerettel rendelkezél, így nem szükséges a blokkok ábráját bemutatni. Közösén készítsétek el a fent leírtaknak megfelelő blokkokat a tanároddal!

Az egyetlen olyan blokk, amely eddig még nem szerepelt, de most szükséges lesz a sprite visszapattanását szabályozó: a blokk hatására a sprite, amely a paraméterében szerepel, úgy fog viselkedni, mint egy labda. Amennyiben a LED kijelző szélén található, akkor visszapattan. Hogy milyen irányban, az a kijelző szélére érkezéstől függ. A kijelző széle falként viselke-



Klasszikus, karaktergrafikával megvalósult teniszjáték. Forrás: https://www.gamasutra.com/view/feature/3900/the_history_of_pong_avoid_missing_.php?print=1

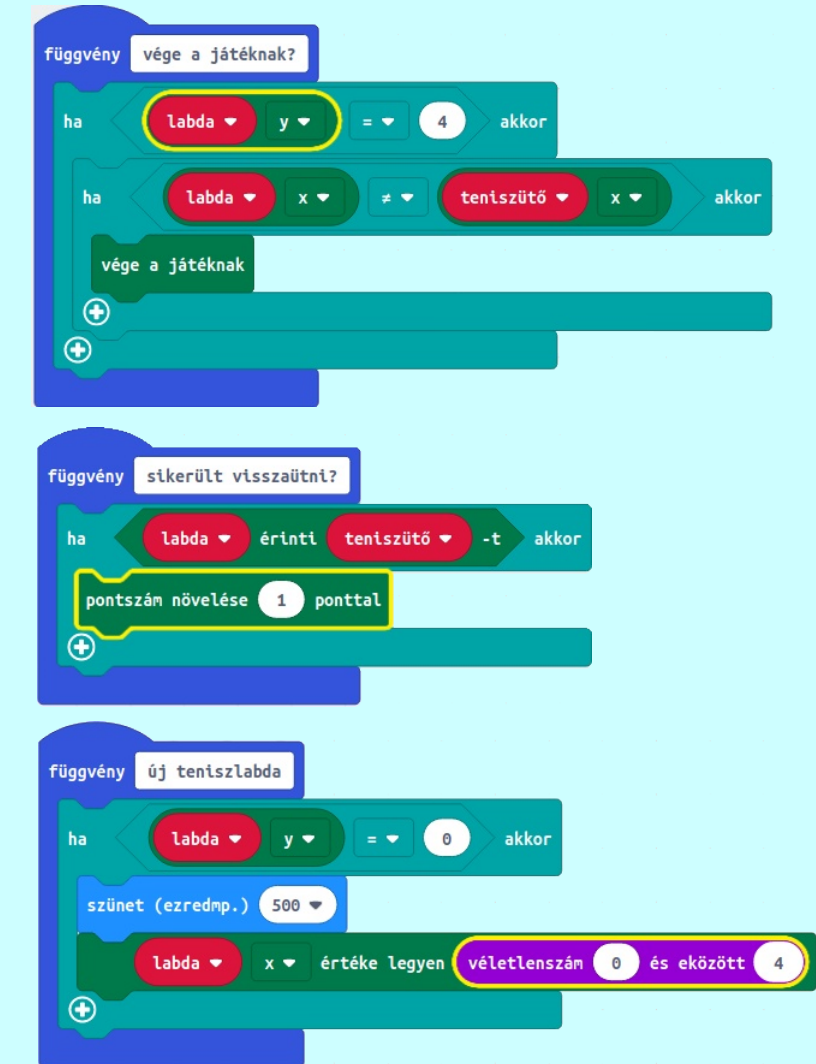
dik. A beérkezés iránya és a visszapat-tanás iránya szimmetrikus a falra a beérkezés pontjában beállított merőlegesre, pont mint a fény tükröződésekor.

A függvények megírása

Maradt három megírandó függvényünk. Ezúttal a jobboldalon szerepelnek a függvények blokkjai.

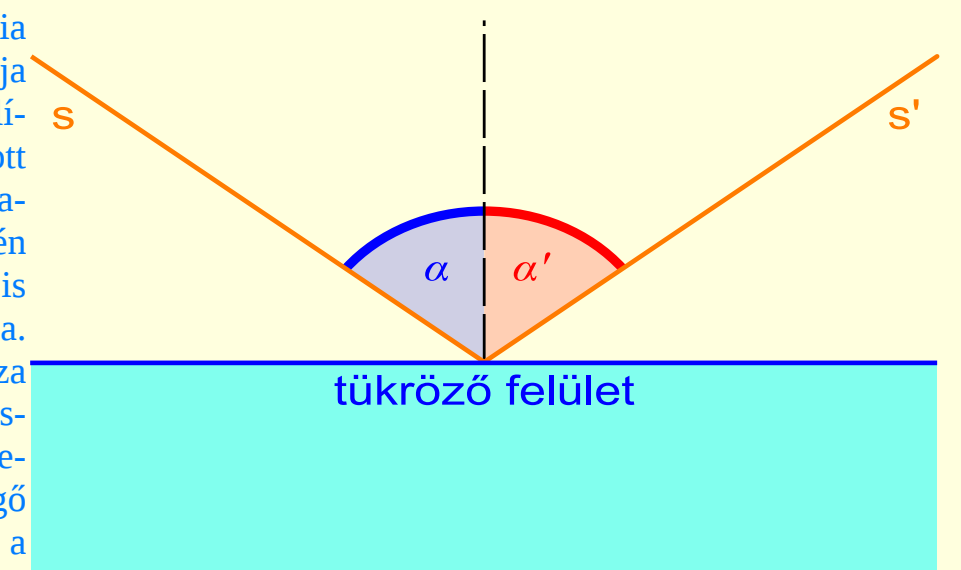
Felmerülhet persze a kérdés, hogy a labda leesését és a visszaütés sikerességét nem lehetne-e egyszerre vizsgálni. A válasz: lehetne, hiszen mindkettő akkor telje-

sül, ha a labda a legalsó sorba ért. Elég lenne ebben az esetben azt megnézni, hogy találkozott-e a teniszütővel. Ha igen, akkor vissza kell pattannia, a pontszámot növelve, ha nem, akkor vége a játéknak.



Egy labda sík felületről történő visszapattanásának módja nagyon hasonlít a fény visszaverődésének módjára. Legalábbis alaphelyzetben. Mint a Wikipédia Fényvisszaverődés szócikkből vett ábra mutatja (jobbaldalt), ha a beérkezési pontban merőlegest állítunk a felületre, akkor az ehhez a merőlegeshez húzott beérkezési és visszaverődési (esetünkben visszapattanási) szög megegyezik. Labda visszapattanása esetén ez azért nem szokott igaz lenni, mert a labda forog is valamerre, amely a visszapattanás szögét befolyásolja. Például a teniszjátékosok szándékosan úgy ütik vissza a labdát, hogy az valamerre forogjon, hogy így a visszapattanás módját befolyásolják. Ráadásul a labda repülési útvonalát is befolyásolja a forgása a levegő áramlási viszonyain keresztül. Ezzel lehet például a

labda kanyarodását előidézni, amely annak visszaütését nehezíti az ellenfél számára.



Hogyan lehetne a játékot továbbfejleszteni?

Ötleteljeteke párokban és fejlesszéteke tovább az alkalmazást!

Az elkészült alkalmazásokat mutassátok be egymásnak és próbáljátok ki! Melyik módosított alkalmazás tetszett neked a legjobban?

Ez a játék már elég bonyolult programot igényelt. Már lényeges, hogy az elkészítése előre megtervezetten történjen. Ezért is szerepelt az előző oldali leírásban az algoritmusok ismertetése és csak a függvények esetében a blokkok bemutatása.

Ha már a függvényeknél tartunk, most ezeket csak egyetlen alkalommal kellett használni, így felmerül a kérdés, hogy mi értelme volt a függvények használatának.

A válasz pont az, hogy az elvégzendő feladat előre tervezése során használtuk fel azt a lehetőséget, hogy nem minden tevékenységet valósítottunk meg egyszerre. A függvényeket ebben az esetben arra használtuk, hogy későbbre halasszuk a gondolkodást arról, a függvény által végrehajtandó tevékenységet hogyan is kell végrehajtani.

Ezzel valójában az egyik leggyakoribb szoftverfejlesztési módszert, a felülről-lefelé tervezés elvét alkalmaztuk. Ez azt jelenti, hogy először nagy általánosságban tervezzük meg a program működését, amikor a tényleges tevékenységek meghatározásával még nem foglalkoznunk – azok meghatározását későbbre halasztjuk.

Amikor már látható nagy vonalakban, hogy hogyan fog a program működni, akkor egy szinttel lejjebb lépve a csak megnevezett tevékenységek algoritmusát tervezzük meg. Ez utóbbi volt az a pont, amikor a három függvényt elkészítettük. Bonyolultabb programok esetében még ezen a szinten is előfordulhat, hogy csak absztrakt mondatok szerepelnek valamely tevékenység megnevezésénél, tovább halogatva azok tényleges megvalósítását. Újabb szinteken azután ezek is meghatározásra kerülhetnek – vagy tovább halogathatóak is-

mét. A lényeg, hogy minden szinten a megoldandó problémát egyre kisebb méretű tevékenységek sorozatára bontjuk, így csökkentjük a probléma méretét mindaddig, míg végül minden probléma megoldható méretűvé csökken.

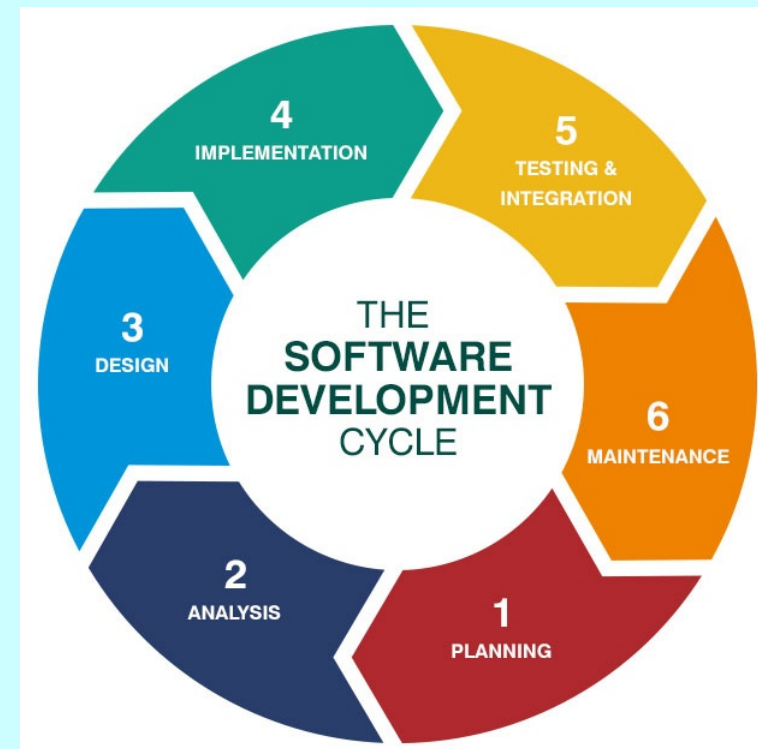
Ennek persze az eredménye lehet olyan függvény, amely újabb résztevékenységeket megoldó függvényeket hív meg.

A felülről lefelé fejlesztés módszerén kívül még többféle módszer létezik. Mindnek megvan a maga helye és felhasználási területe. Ezekkel tanulmányaid során még fogsz találkozni.

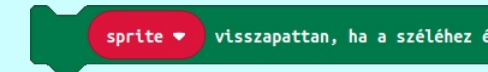
Most már sokféle funkciót ismersz a játékok készítéséhez. Milyen más játékokat lenne érdemes megvalósítani? Ötleteljeteke párokban!

Írjátok le a játékokotok lényegét, mi jelenne meg a kijelzőn, mi a játék menete, szabályai, hogyan lehet irányítani az egyes objektumokat, stb.

A következő alkalommal meg is valósíthatjátok az ötleteiteket.



Újonnan megismert blokkok



Amennyiben a megadott sprite a kijelző szélén tartózkodik éppen, akkor ennek a blokknak a hatására visszapattan onnan, mint egy rugalmas labda. A visszapattanás a tükröződés szabályai szerint történik.

A számítógépes játékok – vagy elterjedtebb nevükön a videójátékok – fejlesztésébe tekinthetél bele egy kicsit az elmúlt néhány foglalkozás folyamán. A meteor-esős játék, illetve főleg a tenisz játék a micro:bit szintjén jól modellezte, hogyan is működik egy ilyen videójáték.

A különbség ezen egyszerűnek mondható játékok és a komoly grafikát felvonultató, erős számítógépet vagy játékgépet igénylő játékok között leginkább a bonyolultságukban van.

Egy komoly videójáték elkészítése több éves fejlesztőmunka eredménye szokott lenni és általában nagyobb fejlesztőcsapat munkája szükséges hozzá. De a játékot megvalósító szoftver felépítése eléggé hasonlít arra, mint amit például a tenisz játék felépítése mutat.

A legelső videójátékokat még az 1960-as években fejlesztették, általában nem kereskedelmi célokra. Akkoriban még személyi számítógépek nem léteztek, hanem nagy méretű, általában tudományos célokra szolgáló számítógépeket használtak. Mégis voltak olyan programozók, akik szabadidejükben egyszerű játékok fejlesztésével foglalkoztak. Ilyen játékokra az előzőekben már láthattál néhány példát. Figyelembe véve a lehetőségeket, abban az időben egy-egy játékprogram megírására általában egyetlen ember is elegendő volt. Mára viszont a számítógépek tudása olyan mértékben fejlődött, hogy az akkori játékokat nagyon kezdetlegesnek éreznénk, sokkal kifinomultabb játékokat várnak el manapság az emberek, amelyek elkészítése nagyon sok részből tevődik össze, így ritkán elegendő egyetlen programozó az elkészítésükhöz.

Manapság egy játék elkészítése egy 20–100 fős csapatot igényel, amelyben a munka különböző részeire szakosodott szakemberek találhatóak: tervezők, művészek, programozók és tesztelők például.

Maga a játék is több elemből tevődik össze.

Egyrészt a játéktól elvárunk magas minőségű grafikát. A programozók általában saját maguk nem képesek a szükséges színvonalú grafika megvalósítására, ezért kell a fejlesztőcsapatba több művész is. Ezek a művészek is több csoportba sorolhatóak:

- grafikusok, akik vázlatrajzok formájában megtervezik a játék szereplőit, az egész játék fő felületét, a grafikai elemek stílusát

- 3D játék esetén a modellezők, akik elkészítik azokat a 3D modelleket, amelyek majd a játékban megjelennek. Ilyen elemek nem csak a játék szereplői, de a díszletek is, amelyek között a játék zajlik

- animátorok, akik a szereplők animációit elkészítik

A másik nagyobb kategóriába tartoznak maguk a programfejlesztők, akik a játék mögötti programkódot valamilyen eszközzel előállítják. És ezen a ponton érünk el oda, hogy az előző játékok ezeknek a játékoknak a működési módját nagyon jól bemutatták.

1. A játékoknak ugyanis mindig van valamilyen kezdeti állapotuk, amelyet a micro:bit esetében az „induláskor” blokkba tett tevékenység valósított meg. Ebben a részben történik meg a játék belső logikájának a felépítése, a nyitó képernyő megrajzolása, a szereplők belső tulajdonságait tároló változók (általában bonyolult adatszerkezetek) létrehozása. Ezt nevezhetjük *inicializáló műveletsornak* is.

2. A belső játékmotor ezt követően bizonyos műveleteket folyamatosan ismételtetni fog. Ennek felel meg a „folyamatosan” blokk tartalma. Ide kerül minden belső állapot figyelése, automatikus tevékenységek végrehajtása, az eközben kialakult események kezdeményezése.

3. A belső események, illetve a játékos által végzett tevékenység hatására bekövetkező események kezelése a harmadik fontos terület, amelyhez különböző műveletek végrehajtása tartozik. Ezeket a micro:bit esetében a Bemenet kategória eseményvezérlő blokkjaiba rakott tevékenységek jelentik.

A játékfejlesztés számára több különböző fejlesztő-szoftver áll rendelkezésre, amely a fenti komponensek – és még további, itt fel nem sorolt komponens – összefogására és egységes kezelésére születtek.

Tipikus példa ilyen játékfejlesztő szoftverre a Unity vagy az Unreal Engine. Ezeknek van egy színterkesztő része, amelyen keresztül a grafikai elemeket lehet összeállítani és van egy játéklogikai része, ahol a programozói feladatok végezhetőek el. A kettő közötti kapcsolatot is képesek a szoftverek kezelni: például a

színterkesztőben levő objektumhoz tulajdonságként lehet hozzárendelni a különböző animációkat vagy a kiváltott eseményeket, amelyeket azután a programozói részen belül lehet lekezelni. Visszafelé: egy programbeli függvényhívással lehet kezdeményezni például egy adott animáció lejátszását.

A fentebb említett programelemek közül a Unity egyébként a 2-es és 3-as pontot egységben kezeli egy olyan ciklust kínálva, amelybe beleírható minden folyamatosan ismétlődő tevékenység, beleértve az egyes események bekövetkezésének ellenőrzését és az események bekövetkezésekor végrehajtandó függvények megírását.

Ebben tehát a játékok és a mikrovezérlők nagyon hasonlítanak egymásra: ugyanannak a kódnak a végrehajtását ismételtetik egyfolytában. A programnak eleje és vége nem sok van, ám a fő tevékenység mellett bemenő eseményekre is figyelnek és azokra a szükséges műveleteket szinte azonnal elvégzik, amikor azok bekövetkeznek.

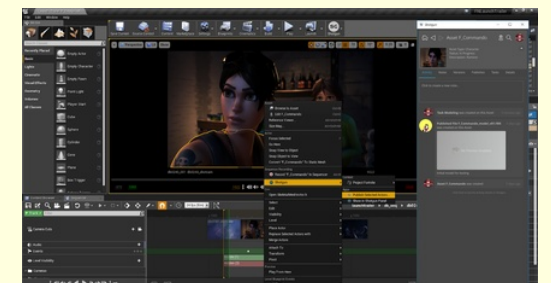
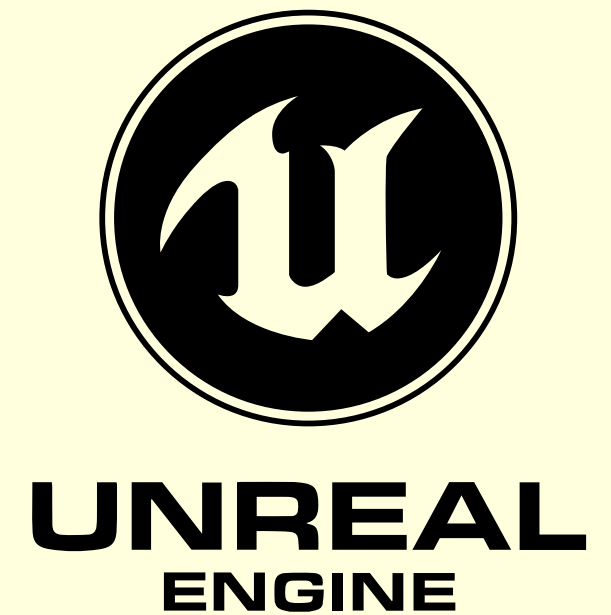
Ezzel szemben a hagyományosnak tekintett számítógépes programoknak mindig van egy indulópontjuk és egy befejezési pontjuk. A micro:bit nyelvére lefordítva ez úgy képzelhető el, mintha csak a „induláskor” blokkot használnánk, az „állandóan” blokkot és az eseményvezérlő blokkokat nem. Ám a mai, grafikus felület használó programoknál már ismét szükséges legalább az események kezelése, de a háttérfolyamatok futtatására továbbra is csak korlátozott mértékben van lehetőség.

Így nézve, a micro:bit, mint minden mikrokontroller eléggé eltér a hagyományos számítógépektől, több lehetőséget biztosít azoknál. Viszont az iparban a mikrokontrollerek használata legalább annyira elterjedt, mint a számítógépeké. Sőt! A legtöbb automatizálási folyamat nem is számítógépet, hanem mikrokontrollert igényel.

Gondoljunk csak az önvezető autókra! Ezekben nem egy olyan programra van szükség, amelyet az indítókulcs elfordításával elindítunk valahol és szépen, előre meghatározott sorrendben hajt végre bizonyos műveleteket. Olyan programra van szükség, amely az autó különböző fizikai állapotait figyelő szenzoroktól,

valamint a külső környezetből érkező információk alapján hoz azonnali döntéseket. Ez tipikusan eseményvezérlést jelent. Közben az autónak haladnia is kell, azaz minden mástól függetlenül adagolni kell az üzemanyagot a motorba, ami háttérben ismétlődő folyamatot igényel.

Nem véletlenül kerültek szóba a külvilág fizikai jellemzőit mérő eszközök, mint bemeneti események forrásai. Az ezen jellemzőket gyűjtő eszközökről, a szenzorokról lesz szó a következő alkalmakkor sorra kerülő feladatokban...



A micro:bit LED-jei nem csak kijelzőként működnek, hanem fényérzékelőként is. A Bemenet kategóriában a „fényerősség” érték-blokk használatával tudjuk lekérdezni a fényerősség szintjét. Ez 0 és 255 között mozog attól függően, hogy teljes sötétséget vagy erős fényt érzékel az eszköz.

Vigyázz, ne keverd össze a LED kategóriában levő, ugyanilyen nevű blokkal! Ez a környezetnek a fényérzékelő által mért fényessége, míg a LED kategóriában levő azt az értéket adja vissza, amilyen fényerővel világítanak a gyújtott LED-ek.

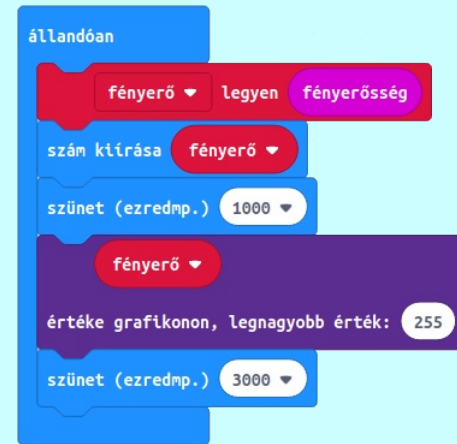
Angolul nem lenne ilyen könnyen összekeverhető a két blokk, mivel a Bemenet kategóriában ez a blokk a „light level” nevet viseli, míg a LED kategória blokkja a „brightness” nevet.

De hogyan tudjuk ezt szemléltetni? Például úgy, hogy a fényerősség értékét megjelenítjük számként, illetve egy grafikonon is ábrázolhatjuk.

Van egy blokk a LED kategóriában, amellyel egy grafikonfélét lehet megjeleníteni, a neve magyarul: „... értéke grafikonon,...”, angolul sem egyszerű a neve: „plot bar graph of ...”. He-lyesen lefordítva talán ez lehetne: „rajzold ki oszlopdiagramon n értékét, ahol m a legnagyobb lehetséges érték”. Persze ez még hosszabb lenne, de leírja a blokk jelentését, ahol n és m a két paraméter értéke ebben a sorrendben.

Azaz a blokk az első paraméter értékét egy oszlopdiagramon ábrázolja úgy, hogy a legnagyobb oszlop méretet a második oszlopban adjuk meg. Valójában a művelet eredményeként annyi százaléka lesz a LED-eknek felkapcsolva, amennyi százalékat az első paraméter kiteszi a második paraméternek.

Készítsd el a 9.1. Fényerősségmérő ábrán látható programot! Takard le a kezeddel a micro:bitet. Ezután világítsd meg például lámpával! Figyeld közben a kijelzőn megjelenő számot illetve utána a grafikon!



9.1. Fényerősségmérő

Teremts kapcsolatot a fény és a hang között! Készíts olyan alkalmazást, amely a fényerősség változását hanghatással, vagy a hang tempójának megváltoztatásával szemlélteti. Ha nincs lehetőség fülhallgató használatára, akkor a szimulátorban teszteld a munkádat! A fényerősséget a szimulátor bal felső sarkában tudod változtatni úgy, hogy a sárga területet megnöveled az egérrel.

A fényerősség méréséhez hasonlóan a micro:bit hőmérsékletet is tud mérni. A Bemenet kategóriában a hőmérő használatához a „hőmérséklet” blokk tartozik. Ezzel lehet az eszközt körülvevő hőmérsékletet megmérni – szerencsénkre az általunk használt C° mértékegységet használva.

Alakítsuk át korábbi alkalmazásunkat úgy, hogy a hőmérsékleti érték és annak grafikonja változzon a kijelzőn! Állítsuk be a maximális értéket 30-ra (vagy ha nyári meleg van, akkor egy kicsivel nagyobbra)!

Ugye nem szükséges megadni a projekt blokkábráját...?

Készítsünk olyan eszközt, amelyet a vak emberek is használhatnak a hőmérséklet lekérdezésére. Ilyenkor hanggal tudnánk szemléltetni a hőmérsékleti értéket. Ötleteljünk, hogyan lehetne ezt megcsinálni?

- Annyiszor lejátszunk egy hangot, ahány fok van?

Hátránya, hogy sokat kell számolnia a vak embernek! Jobb lehet, ha külön lejátszuk a 10-es helyiértéken lévő számot és külön az 1-es helyiértéken lévő. Így például a 23 fokot le lehet úgy játszani, hogy kettőt pittyeg az eszköz, majd egy kis szünet után (esetleg kicsit más hangmagassággal) hármat.

A feladat végén a második megoldás megvalósításához meg kell határozni, hogy az adott hőmérsékleti értékben hányszor van meg a 10 maradék nélkül, illetve mennyi a maradék.

Ehhez nyilvánvalóan a Matek kategória megfelelő blokkjait kell használni.

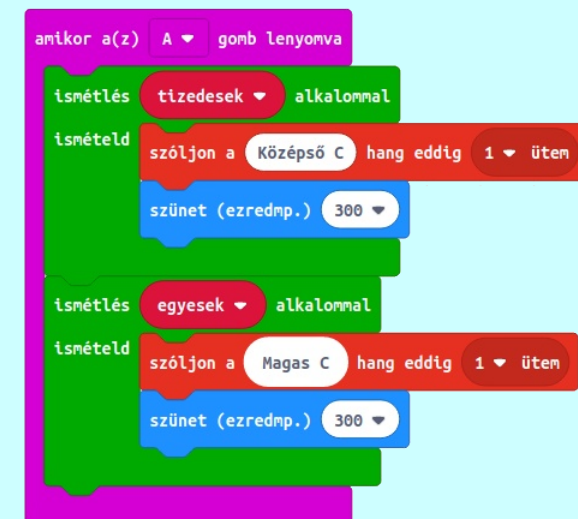
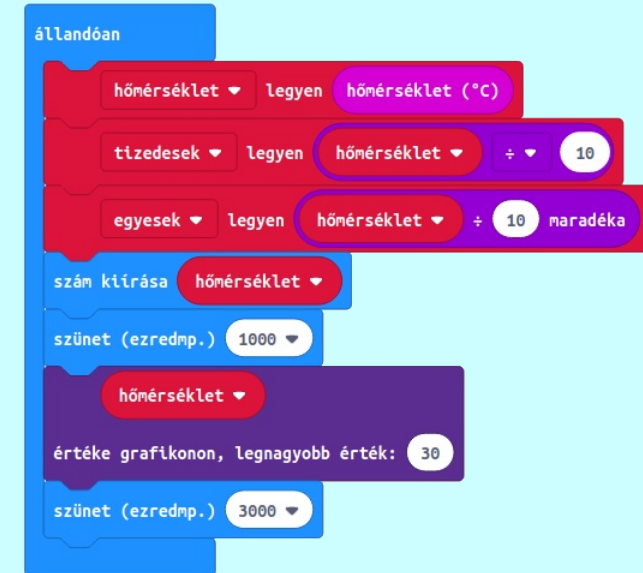
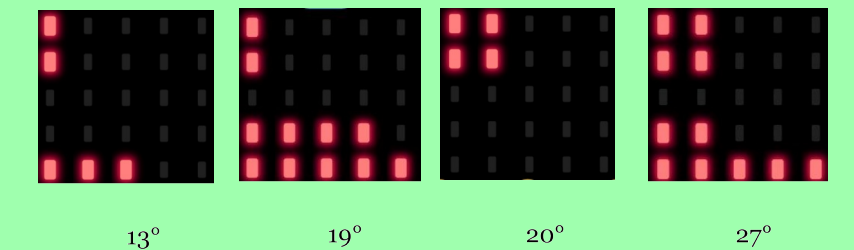
Az előző projekthez adjuk hozzá a hanghatást úgy, hogy az ‚A’ gombot kelljen hozzá megnyomni!

A beépített grafikon leolvasása eléggé nehézkes. Készíts olyan grafikon, amely a következőképpen ábrázolja a számokat:

A felső két sorban ábrázoljuk a tizedes helyiértéken lévő számokat, két pont magas pálcikákkal. Az alsó két sorban pedig az egyes helyiértékeknek megfelelő számító pontot gyűjtjük ki.

Ez a grafikon akkor jelenjen meg, amikor a ‚B’ gombot nyomjuk le.

Íme néhány példa:



A micro:bit egy iránytűt is tartalmaz. A Bemenet kategóriában található ehhez egy blokk, amellyel az általa mutatott szög értékét tudjuk lekérdezni. A blokk által visszaadott szögérték 0 és 359 közötti lehet.

iránytű iránya (°)

Az iránytű a Föld mágneses mezejét használja fel az égtájak megmutatására. Földrajzi ismereteidből már bizonyára ismerősek az égtájak. Az iránytű elméletileg az Északi irányt mutatja, így ezt szokás 0° iránynak tekinteni. Azért elméletileg, mert egyrészt a Föld északi mágneses pólusa nem esik teljesen egybe a geológiai északi pólussal, vagyis a mágneses északi irány nem teljesen egyezik meg a földrajzi északi iránnyal, az iránytű pedig az előbbi irányba mutat. Másrészt a közelben levő mágneses mezők megzavarhatják az iránytűt, amely így a valódi északi pólustól eltérő irányba mutathat.

Az Északi irányhoz képest, ha felülről nézünk rá az iránytűre, akkor az óramutató járásának irányába növekvő szögekkel szokás jelezni a többi irányt. Így még Észak értéke 0°, Kelet 90°, Dél 180° és Nyugat 270° értéknek felel meg.

Készítsük el közösen a következő alkalmazást: Az „A” gomb megnyomásakor tároljuk el változóban a szög értékét, majd írassuk ki a képernyőre.

Az első futtatásnál a micro:bit kalibrálni akarja az iránytűjét. Ilyenkor megjelenik a „Draw a circle” szöveg a kijelzőn. Ekkor az eszközt különböző irányokba kell dönteni úgy, hogy a kijelzőn egy körvonal jelenjen meg. Ezután fog működni az iránytű.

Bármikor ezután is rávehető az eszköz az iránytű kalibrálására a Bemenet kategória haladó részében található iránytű-kalibráló blokkal is.

A szimulátorban a kalibráció elmarad, hiába használjuk ezt a blokkot is. A szimulátort ugyanis nem szükséges kalibrálni: az iránytű bemenetét a robotfejnél megjelenő nyilacska mozgatásával lehet szimulálni.



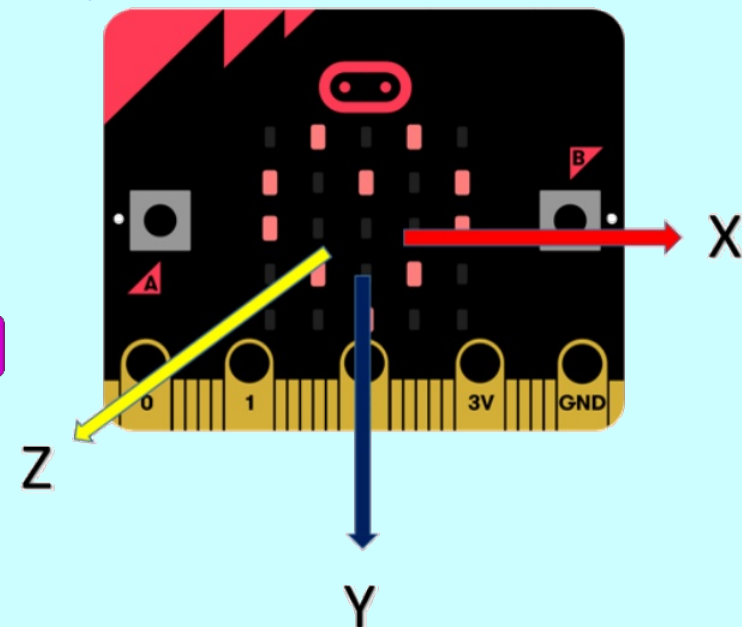
Fejleszd tovább úgy az alkalmazást, hogy ha a „B” gombot lenyomjuk, akkor a következő történjen:

- Ha a szög 0 és 45 között van, vagy 316 és 359 között, jelenjen meg egy felfelé mutató nyíl (ez jelképezi az Északi (North) irányt).
- Ha a szög 45 és 135 között van, jelenjen meg egy jobbra mutató nyíl (ez jelképezi a Keleti (East) irányt).
- Ha a szög 136 és 225 között van, jelenjen meg egy lefelé mutató nyíl (ez jelképezi a Déli (South) irányt).
- Ha a szög 226 és 315 között van, jelenjen meg egy balra mutató nyíl (ez jelképezi a Nyugati (West) irányt).

A nyilakat nem kell megrajzolni: az Alapok kategória tartalmaz egy blokkot nyilak megjelenítésére...

nyíl megjelenítése: Észak

A szenzorok (érzékelők) között gyorsulásmérőt is találhatunk. Ez a micro:bit mozgatására képes reagálni – bizonyos határok között. Mégpedig külön-külön a három irányra.



Az ábra az előző hasáb alján mutatja a három irányt, amelyben a gyorsulásmérő mér.

Készítsünk közösen egy alkalmazást, amely az X irányú gyorsulás mértékét jelzi ki grafikonnal! Mivel a gyorsulás lehet negatív is, vegyük az érték abszolút értékét a kijelzéshez. A maximum érték legyen 2000, ami alapértelmezésben 2 g gyorsulásnak felel meg.

A gyorsulás mérésére használható blokk is a Bemenet kategóriában található.

Amennyiben az X irányú gyorsulás (mg) x elmozdulás mérését állítjuk be, akkor ha kizárólag függőlegesen, vagy az eszköz felületére merőlegesen mozdítjuk el, akkor nem tapasztalunk változást.

Változtasd meg a kódot úgy, hogy az x helyett az y, illetve a z irányú gyorsulást mérje és próbáld ki az alkalmazást!

Miért mutat y irány esetén nyugalmi helyzetben is gyorsulást az eszköz? Mennyi gyorsulást mutat?

Mikor méri az eszköz a magad felé mozgás gyorsulását?

Amennyiben nem az egyes irányokba eső gyorsulást akarjuk mérni, hanem egy összesített eredményre vagyunk kíváncsiak, akkor a „gyorsulás” blokk listájában az erősség (strength) értéket kell választani. Próbáld ki ezt a változatot is!

A szabadon eső testek gyorsulása egyenletes, az úgynevezett nehézségi (gravitációs) gyorsulás jele g . Normál értéke $g = 9,81 \text{ m/s}^2$ körüli. A tényleges érték a földrajzi helyzettől és a felszíni magasságtól függ.

A micro:bit a g ezred részében méri a gyorsulást.

gyorsulásmérő tartománya legyen 1g

A „gyorsulásmérő tartománya legyen...” blokk segítségével ezt az alapértelmezett skálát lehet átállítani 2,... 8 g-re. A beállított érték lesz a legnagyobb mérhető érték.

Ide kellene valami érdekesség az iránytű használatáról és jelentőségéről, illetve a Föld mágneses teréről...

Újonnan megismert blokkok

fényerősség

Az eszközt körülvevő térben jelen levő fényerősséget adja vissza a [0..255] intervallumba eső értékkel: 0 = teljes sötétség, 255 = teljes megvilágítás.

gyorsulás (mg) x ▼

Az eszközre ható gyorsulás értékét adja meg a nehézségi gyorsulás (g) ezredrészében (lásd még a „gyorsulásmérő tartománya legyen...” blokkot is).

A legördülő listában az x, y, z koordináták állíthatók be, vagy ezek összegzéséből az „erősség” értéket lehet használni.

iránytű iránya (°)

Az iránytű irányát adja vissza fokban, Észak felől indulva, az óramutató járásával megegyező irányban [0..359] intervallumba eső fok értékekkel. Első használata előtt kalibrálni kell (automatikusan megtörténik).

hőmérséklet (°C)

Az eszköz által mért környezeti hőmérsékletet adja C°-ban.

mágneses térerő (μT) x ▼

A mágneses térerő értékét adja vissza x, y, z tengely szerint, vagy az ezekből számított erősségként.

dőlési szög (°) bólintás ▼

Megmondja fokban, hogy a micro:bit mekkora szögben van megdöntve. A két beállítható érték a listában a vizsgált döntés irányát határozza meg: bólintás: fel/le; döntés: balra/jobbra.

iránytű kalibrálása

Az iránytű kalibrálását lehet vele végrehajtani. Amennyiben a program végrehajtása során kell az iránytű értékét lekérdezni és még nem történt meg, akkor automatikusan megtörténik a kalibrálás a program indulásakor. Ezt követően bármikor újra lehet kalibrálni az iránytűt ezzel a blokkal.

gyorsulásmérő tartománya legyen 1g ▼

A gyorsulásmérő mérési tartományát lehet ezzel a blokkal beállítani 1g, 2g, 4g vagy 8g értékre. A beállított érték lesz a legnagyobb érték, amelyet mérni tud a gyorsulásmérő. A mérés azonban mindenképpen a nehézségi gyorsulás ezredrésze pontossággal történik.

9. Ismerkedés a szenzorokkal

0

értéke grafikonon, legnagyobb érték:

0

Egy oszlopdiagramot rajzol ki a LED mátrixon. Az oszlop lehetséges maximális mérete adandó meg a második paraméterrel, míg az oszlop tényleges mérete az első paraméterrel. Ez alapján a LED-eknek annyi százaléka fog világítani, ahány százaléka az ábrázolni kívánt érték a maximum értéknek.

A blokk a LED kategóriában található.

nyíl megjelenítése:

Észak ▼

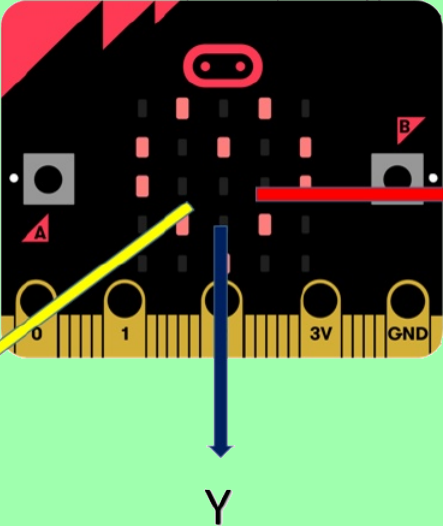
Különböző égtájak felé mutató nyilakat jelenít meg a kijelzőn.

Sorrendben: Észak, Észak-kelet, Kelet, Dél-kelet, Dél, Dél-nyugat, Nyugat, Észak-nyugat a nyilak iránya.

A blokk az Alapok kategóriában található.

A múlt alkalommal megismerkedtünk a szenzorokkal, most haladóbb alkalmazásokat készítünk a szenzorok által mért adatok felhasználásával.

Készíts olyan alkalmazást, amely a micro:bit x irányú gyorsulását a LED-ek fényerejének módosításával jeleníti meg!



Indításkor minden LED legyen kigyújtva. A háttérben futó folyamatban egy változóban tárold el az x irányú gyorsulás abszolút értékét, a fényerőt pedig ezen érték ötödére állítsd be!
Próbáld ki az alkalmazást úgy, hogy oldalirányba mozgatod az eszközt!

Most egy olyan alkalmazást készítünk, amely számolja az ugrásainkat. Ennél a feladatnál is a gyorsulásmérő által mért értékeket fogjuk felhasználni.

Az elkészítendő program blokk-kódját mutatja a 10.1. ábra.

Az alkalmazás lényege, hogy folyamatosan vizsgáljuk az Y irányú (tehát függőleges) gyorsulás értékét, és ha az meghaladja a 2000-es értéket, akkor egyel növeljük az ugrások számát és megjelenítünk egy vigyorgó fejet. Az ‚A‘ gombbal ki tudjuk írni az ugrások számát, az ‚A+B‘ gombokkal pedig le tudjuk nullázni az ugrás értékét.

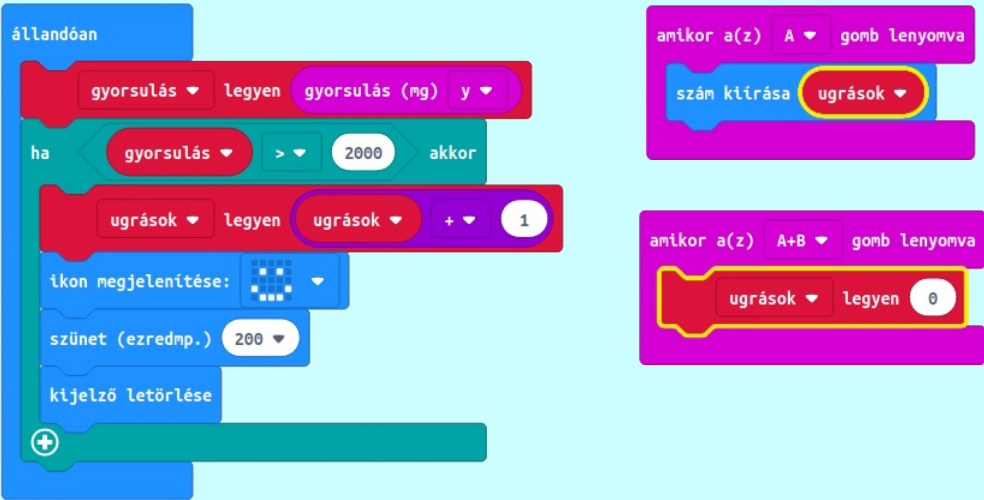
Próbáljuk ki az alkalmazást!

Hogyan lehetne továbbfejleszteni az alkalmazást? Alakítsatok párokat és ötleteljetek! Valósítsátok meg az ötleteiteket!

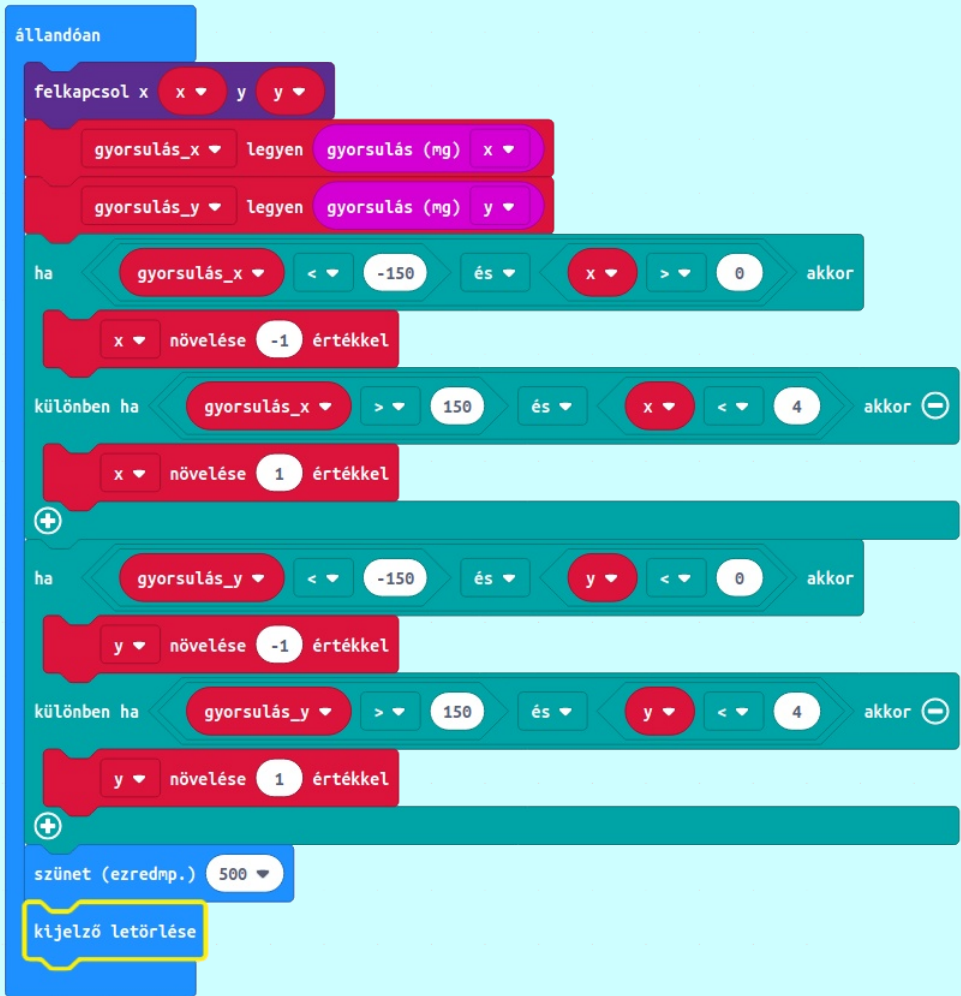
Biztosan ismered azt az ügyességi játékot, amelyben egy kanálon kell elhelyezni egy tojást és úgy kell teljesíteni egy akadálypályát, hogy a tojás ne törjön össze. Most valami hasonlót fogunk készíteni micro:bit-re hangolva.

Próbáld megfejteni, mit csinál a 10.2. ábrán látható kód!

Próbáljuk ki az alkalmazást! Amennyiben a micro:bit-ek elemtartóval is rendelkeznek, a számítógéptől is el lehet távolodni. Ebben az esetben a teremben készíttetek egy rövid szlalom pályát, amelyet úgy kell teljesíteni, hogy a pont a kijelzőn ne érje el a szélső sorokat, oszlopokat.



10.1. ábra: az Ugrás számláló program blokkjai



10.2. ábra

Most már többféle alkalmazást is láttatok a szenzorok használatára. Párokbán ötleteljetek, hogy milyen alkalmazást tudnátok készíteni, amelyben szerepet kapnak a szenzorok. Ez lehet játék vagy más alkalmazás is.

A micro:bit-ek képesek egymással kommunikálni rádiókapcsolaton keresztül. Ehhez az kell, hogy azonos csoportba szervezzük őket a Rádió kategória „rádió: csoport legyen n” (angolul: „radio set group”) blokkjával.

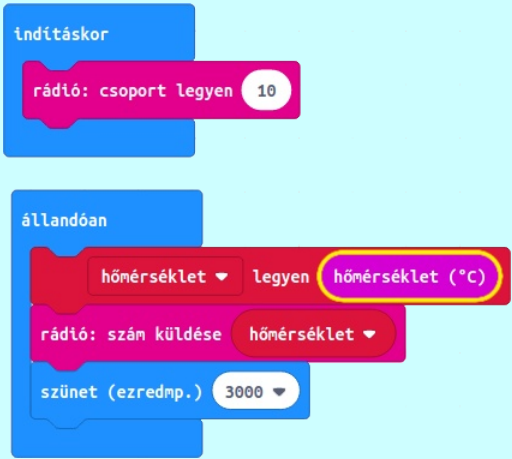
Az itt az n helyén megadott szám lesz a csoport azonosítója. Ha két micro:bit esetében ugyanaz az érték kerül beállításra, akkor ők kerülnek azonos csoportba. Természetesen kettőnél több eszköz is kerülhet azonos csoportba, de ebben az esetben kicsit jobban oda kell majd figyelni köztük a kommunikáció koordinálására. Az érték 0 és 255 között választható.

A következőkben közösen egy alkalmazást készítünk, amelyben az egyik micro:bit megméri a hőmérsékletet és átküldi a vele egy csoportban levő másik micro:bit-nek, ami pedig megjeleníti azt.

Először készítsük el az adó alkalmazást a 11.1. ábrának megfelelően.

Egy értéket a rádiókapcsolaton keresztül a „rádió: szám küldése” (angolul: „radio send number”) blokkal lehet elküldeni. Ennek a paraméterében kell elhelyezni az átküldendő számértéket – esetünkben a hőmérséklet értékét.

A vevő eszköz számára egy másik program szükséges. Itt ugyanazt a számot (10) kell azonosítónak használni, mint az előbbi esetben. A Rádió kategóriában a fogadáshoz egy eseményvezérlő blokk tartozik. Az eb-



11.1. ábra: A hőmérséklet küldő program blokkábrája

ben elhelyezett tevékenység hajtódik majd végre, amikor az eszköz fogadja a számot.

A fogadó programot mutatja a 11.2. ábra. Mivel két micro:bit-nek kell egymással kommunikálnia, így párokat kell alkotnotok a programok kipróbálásához. Egyikőtök a küldő, a pár másik tagja a fogadó programot tölts fel az eszközére. Minden pár más rádiócsoportot használjon!

Ismét párban kell dolgoznotok. Készítsetek olyan alkalmazást, amely két micro:bit közti kommunikációt valósít meg a következők szerint:

- Az adó alkalmazással lehessen megadni egy tetszőleges 0 és 15 közötti számot és azt gombnyomásra el lehessen küldeni a másik micro:bit számára.
- A vevő alkalmazás automatikusan jelenítse meg a számot, amikor megkapta azt, majd utána jelenítse meg grafikonon.

Továbbra is párokban dolgozva készítsetek olyan alkalmazást, amely két micro:bit közti kommunikációt valósít meg a következők szerint:

- Az adó alkalmazás 5 másodpercenként küldje át azt, hogy milyen fényerősséget mér.
- A vevő alkalmazás a fényerősséget úgy jelenítse



11.2. ábra: A hőmérsékletadatot fogadó program blokkábrája

meg, hogy az összes pont legyen kigyújtva a kijelzőn, viszont a kijelző világosságértéke legyen a másik eszköztől kapott értékkel azonos. (Emlékeztető: a fényerő a [0..255] intervallumba esik, 0 a teljes sötétséget, 255 a teljes fényerőt jelenti.)

Készítsünk közösen egy olyan játékot, amelyet többen lehet játszani! A játék nev: forró krumpli.

A játék leírása: Az egyik micro:bit ,A' gombjának lenyomásával induljon a játék. Amikor megjelenik az ábra (forró krumpli) a kijelzőn, minél gyorsabban meg kell nyomni a ,B' gombot. Ekkor az ábra a másik játékos micro:bit-jére ugrik. Neki szintén minél előbb a ,B' gomb megnyomásával meg kell szabadulnia a forró krumplitól.

Ha valamelyik játékos akkor nyomja meg a ,B' gombot, amikor még nincs nála a forró krumpli, az pontlevonást kap, a sikeres átküldés viszont növeli a pontszámot.

A játék 30 másodpercig tartson. A játék végén jelenjenek meg a pontszámok.

A játékot úgy valósítsuk meg, hogy ne legyen külön adó és vevő, hanem ugyanaz a program fusson mind-egyik micro:bit-en.

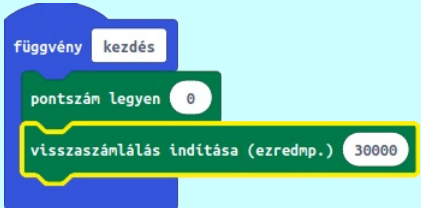
A program indításakor állítsunk be egy közös rádió csoportot a két eszköz számára (11.3. ábra).

A játék kezdetekor be kell állítanunk a pontszámot nullára és el kell indítanunk egy visszaszámlálást 30 másodpercről. Készítsünk erre egy függvényt „kezdés” néven, mivel ezt a kódot többször is fel kell majd használnunk (11.4. ábra).

Készítsünk egy másik függvényt is



11.3. ábra



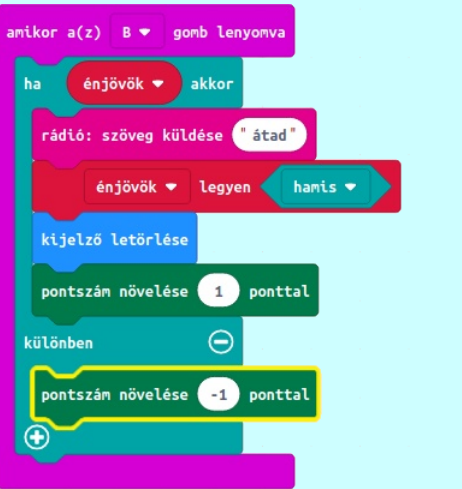
11.4. ábra

„forrókrumpli” néven, amely megjeleníti a kijelzőn az ábrát (11.5. ábra).

Most térjünk át arra, hogy az ,A' gomb lenyomásával indítjuk el a játékot! A kezdési állapotról értesíteni kell a másik micro:bit-et is. E célból küldjünk át egy szöveget, mondjuk „kezdés” tartalommal. Ezek után hívjuk meg a „kezdés” függvényünket, hiszen a pontszámot le kell nullázni és elindul a játék. Várakozzunk picit, majd rajzoljuk ki az ábrát a képernyőre.

Nyilván kell tartanunk azt is, hogy mi következünk-e a krumpli átadásánál. Ehhez vezessünk be egy változót, például „énjövök” néven. A változó értéke legyen igaz (true), hiszen azon a sor, aki a játékot kezdte. A programkód a 11.6. ábrán látható.

Most nézzük, minek kell történnie a ,B' gomb lenyomására! Ezzel a gombbal lehet átküldeni a másik eszközre a forró krumplit. Amikor az „énjövök” változó igaz, akkor tényleg át tudom küldeni az ábrát a másik eszközre, és megnövelhetem a pontszámot. A másik micro:bit-et informálni kell arról, hogy átadom a forró krumplit, ezért át lehet küldeni neki egy „átad” üzenetet. Fontos, hogy átadás után az „énjövök” változó tartalmát át kell állítani hamisra. Egyben a pontszámot egyel meg kell növelni. Ellenben, ha akkor nyomta meg a játékos a gombot, amikor

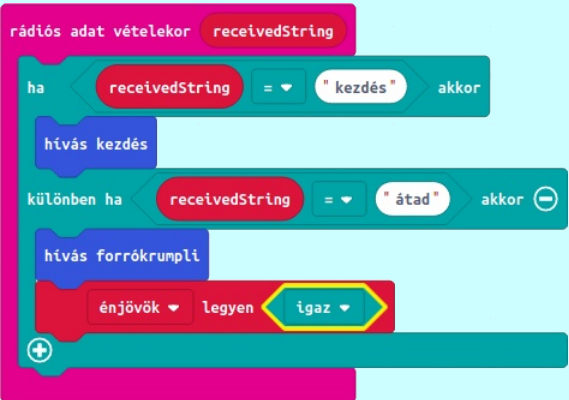


11.7. ábra

nem nála volt a krumpli, akkor le kell vonni tőle egy pontot. A szükséges kód a 11.7. ábrán látható.

Most már csak az egyes üzenetek kezelését kell megoldani.

A kapott értékfől függően vagy a kezdés függvényt kell meghívni, vagy meg kell jeleníteni a forró krumplit. Amikor megkaptuk a forró krumplit, az „énjövők” változót is be kell állítani, hogy tovább lehessen azt adni (11.8. ábra).



11.8. ábra

Ötletjeltek párban arról, hogyan lehetne továbbfejleszteni a játékot, hogy növekedjen a játékelmény. Kezdjétek megvalósítani az ötleteiteket.

Meg lehetne valahogyan oldani, hogy ne csak ketten játszassák a játékot úgy, hogy mindig csak egy valaki kapja meg a forró krumplit véletlenszerűen? Hogyan oldható ez meg?

Ötletjeltek közösen arról, hogy milyen többfelhasználós játékokat lenne érdemes megvalósítani. Jegyezzétek fel az ötleteket a későbbi megvalósításhoz. A megvalósításra kb. 140 perc jut majd a következő két alkalommal, ehhez érdemes mérni a projektek nehézségét.

11. Kommunikáció a micro:bit-ek között

Újonnan megismert blokkok

rádió: csoport legyen 1

A rádiókapcsolat csoportazonosítóját lehet vele beállítani. A beírt számérték lesz az azonosító. Amely micro:bit-ek azonos csoportazonosítóval dolgoznak, azok tudnak egymással kommunikálni. A csoportazonosítónak a [0..255] intervallumba kell esnie.

rádió: szám küldése 0

A blokk egy számot küld rádión. A számot minden olyan közeli micro:bit megkapja, amely ugyanazt a rádiócsoportot használja, mint a küldő.

rádió: adat vételekor receivedNumber

Ez a vezérlőblokk akkor kerül végrehajtásra, ha az eszköz a rádión keresztül egy számértéket kap.

rádió: érték küldése "name" = 0

Egy (név,érték) párt küld az azonos rádiócsoportba tartozó micro:bit eszközökre.

rádió: adat vételekor name value

Amennyiben az azonos rádiócsoportba tartozó eszközök valamelyikétől egy (név, érték) pár érkezik, akkor hajtódnak végre az itt megadott tevékenységek.

rádió: szöveg küldése ""

Egy szöveget küld az azonos rádiócsoportba tartozó eszközökre.

rádió: adat vételekor receivedString

Amennyiben egy másik, azonos rádiócsoportban levő eszközről egy szöveges üzenet érkezik, akkor hajtódnak végre az ebben az eseményvezérlő blokkban elhelyezett műveletek.

Papír zongora

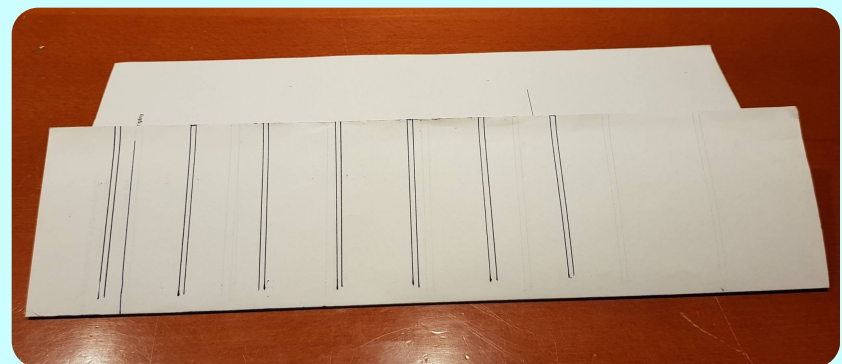
A következőkben egy papírzongora elkészítésének lépései szerepelnek. A fotókat és a leírást Fülöp Tamás készítette (lásd az Irodalomjegyzékben).

Hozzávalók egy zongorához:



- 1 db A4-es lap.
- 4 db alufólia csík (3 db 1 cm széles, 1 db 8 cm széles)
- 4 db krokodilcsipesz vezetékkel
- 1 db fülhallgató csatlakozó vagy krokodilcsipesz a hangszóró/fülhallgató csatlakoztatásához
- 1 db hangszóró/fülhallgató

1. lépés



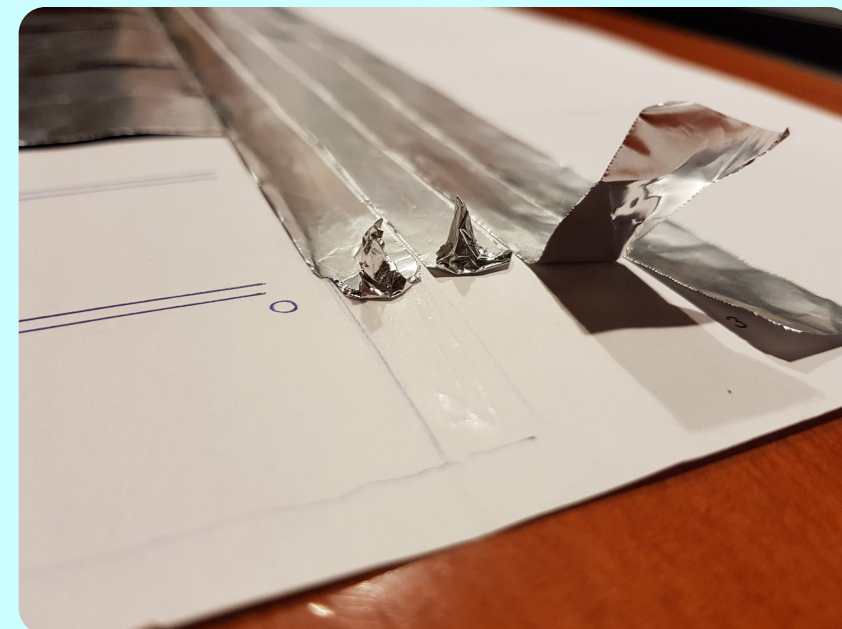
Hajtsuk fel a papírt hosszában. A felhajtott rész max. 7 cm-es legyen. Jelöljük be a 7 billentyűt. A billentyűk között hagyjunk pár milimétert, hogy a lenyomott billentyű a szomszédját ne húzza magával.

2. lépés



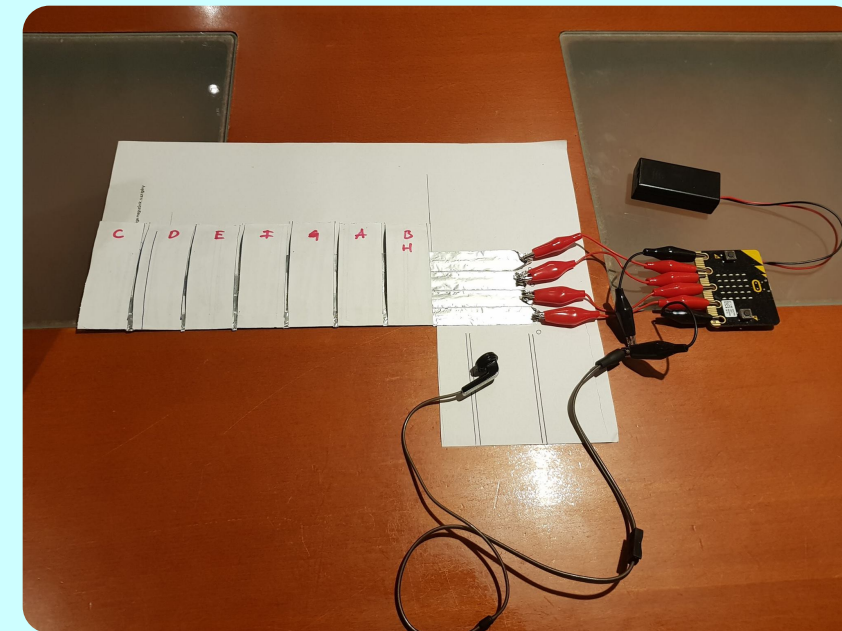
Ragasszuk fel a csíkokat. A vastagot ragasszuk a széleire. Ez lesz a billentyűk alján. A vastag alufóliának túl kell érnie a hajtáson. A csíkok azon végét, amelyek a micro:bit-hez fognak csatlakozni, ne ragasszuk le végig!

3. lépés



Vágjuk le a vastag csíkot méretre. Gyűrjük fel a csíkok végeit. Erre a tüskékre fognak ráharapni a krokodilcsipeszek... vagy ide tudunk egy vezetékét vagy drótot csatlakoztatni.

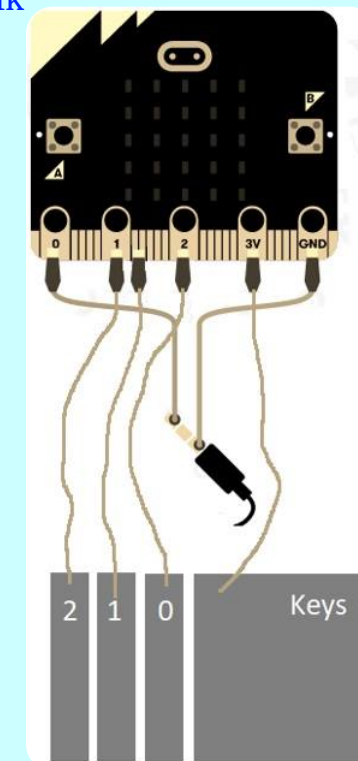
4. lépés



Vágjuk ki a billentyűket (papír és alufólia együtt). Hajtuk rá a billentyűket a csíkokra.

5. lépés

Csatlakoztassuk a micro:bit-et a zongorához és egy fülhallgatóhoz. A harmadik csipesz a pin8-ra csatlakozik.



Az alábbi kódot kell a micro:bit-re tölteni a zongora életre keltéséhez:

<https://makecode.microbit.org/03107-69090-41198-50535> (rá lehet kattintani a böngészőben megnyitáshoz)

Amikor megnyomjuk az „A” gombot, akkor véletlenül választott dallamot játsszon le a micro:bit. Utána próbáljuk ugyanazt a dallamot elzongorázni!

Álljon össze három csoport! Módosítsátok a zongorák hangnemét! Az egyik zongora alacsony, a másik közép, a harmadik magas hangon szólaljon meg. Próbáljátok meg mindhárom zongorán egyszerre ugyanazt a dallamot eljátszani.

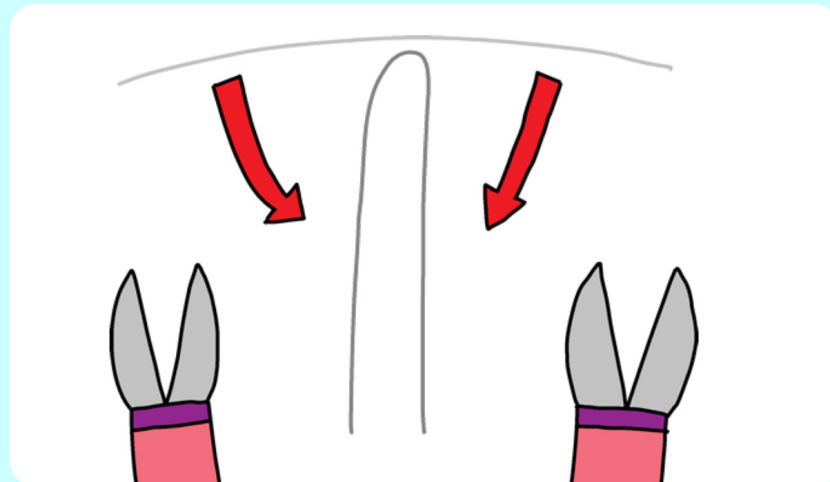
Csak nyugalom!

Ehhez a játékhoz biztos kezek szükségesek. A játék lényege, hogy egy hurkot kell egy vezeték mentén végigvinnünk úgy, hogy nem érhetünk hozzá a vezetékhez. Amennyiben mégis hozzáérnénk, akkor a micro:bit számolni fogja a hibázásainkat.

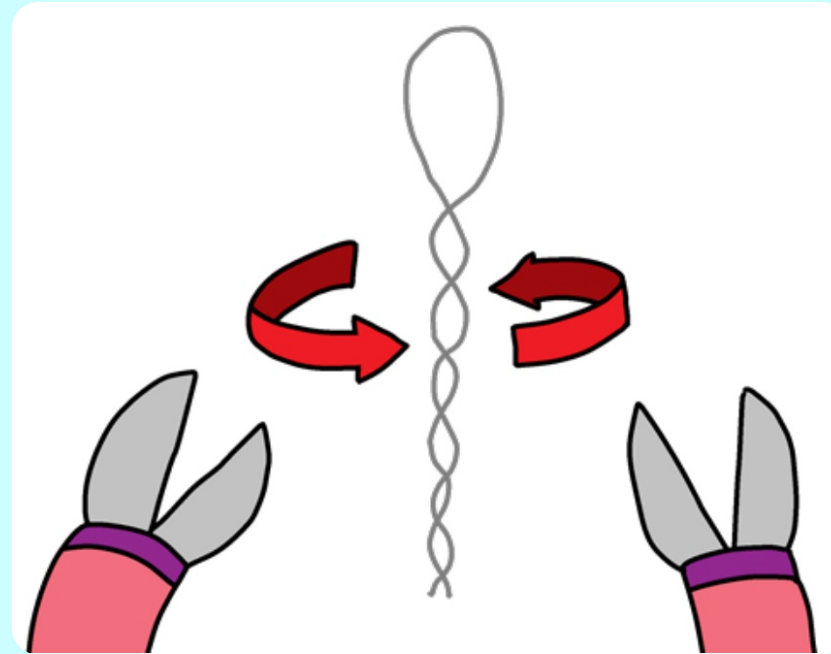
Hozzávalók egy játékhoz:

- Kb. 50 cm hosszú fém drót
- Modell gyurma (vagy valami hasonló anyag, ami nem vezeti az áramot, de megtartja a drótot, pl. hungarocell)
- 10 cm szigetelőszalag
- 2 db krokodilcsipesz

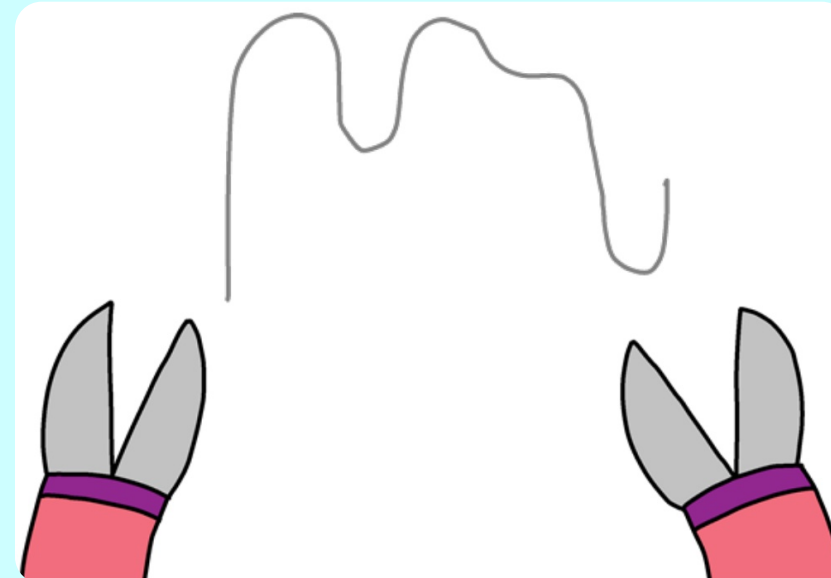
1. lépés



Készítsük el a hurkot: Vegyünk egy kb. 20 cm hosszú drót darabot, majd hajlítsuk meg úgy, hogy a két szár kb. egyenlő hosszúságú legyen. Ezután csavarjuk össze a két szarát.

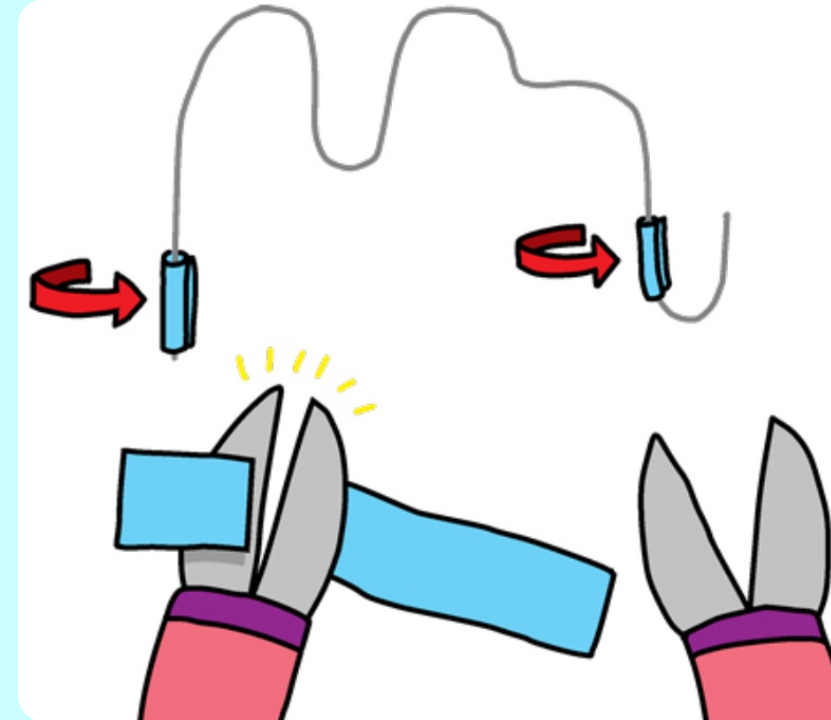


2. lépés



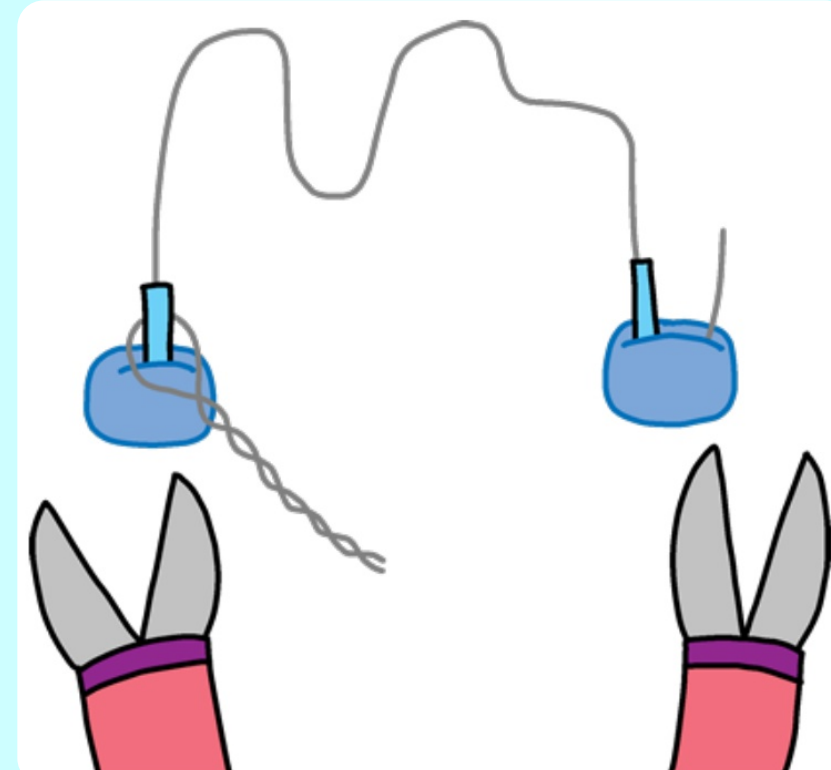
A maradék 30 cm hosszú drótból hajtogassunk egy pályát, e körül kell majd a hurkot végigvezetnünk. A drót egyik végét hajlítsuk felfelé.

3. lépés



A szigetelőszalagot a drót mindkét végén ragasszuk fel úgy, hogy maradjon csupasz drótvég is.

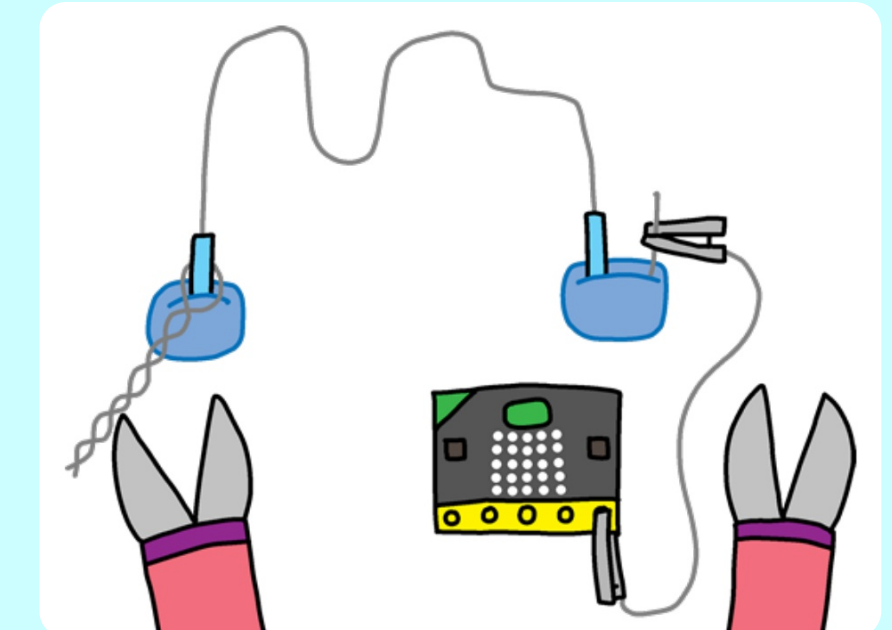
4. lépés



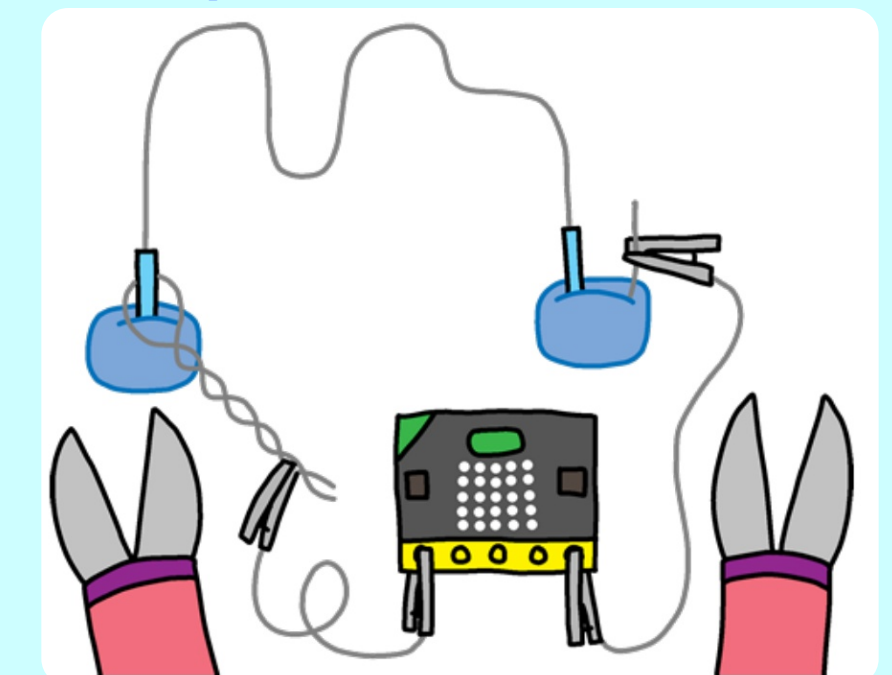
14. Barkácsoljunk!

A hurkot fűzzük át a dróton és a drót két végét helyezzük el a gyurmában (vagy amit használunk).

5. lépés



A micro:bit GND (föld) kivezetését csatlakoztassuk a krokodilcsipesszel a drót oldalához.



A micro:bit P0 lábát csatlakoztassuk a hurokhoz egy krokodilcsipesszel.

6. lépés

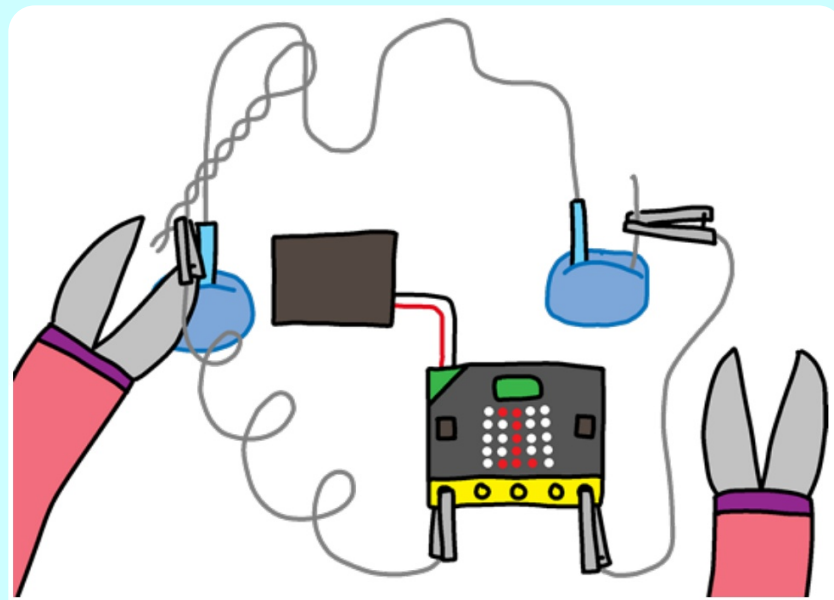
Készítsük el a 14.1. ábrán látható kódot!



14.1. ábra

Amikor hozzáérünk a hurokkal a vezetékhez, akkor meg kell jeleníteni egy nagy X-et a kijelzőn, picit várni, majd növelni a hibák számát egyel és az aktuális hibaszámot kijelezni.

A játék az „A” gomb megnyomásakor indul. A hibák száma ilyenkor lenullázódik és megjelenik egy ábra, amely a pályát jelképezi. Egy kis szünet után ezt felváltja a hibák számának kijelzése (aminek itt még ille- ne nullának lennie).



Készítsetek különböző nehézségű pályákat, majd próbáljátok ki az összes pályát!

Az ebben a leckében szereplő feladatokban az újdonság a Haladó/Csatlakozó lábak kategória. Ebben a kategóriában olyan blokkok vannak, amelyekkel a lábakra csatlakoztatott eszközökkel lehet kommunikálni – akár analóg, akár digitális formában: küldeni és fogadni is lehet a lábakon keresztül üzeneteket. A Zene kategória blokkjai is ezeknek a lábakra egy részét használják a hangok lejátszására.

Amennyiben nem az eszköz kér információt a lábról, hanem a lábon jelenik meg egy kívülről érkező üzenet, akkor viszont a 14.1. ábrán látható „amikor a ... láb lenyomva” nevű eseményvezérlő-blokk tartalma hajtódik végre. Ez a blokk viszont a Bemenet kategóriában található a többi eseményvezérlő blokkhoz hasonlóan.

Ezeknek a lábakra a segítségével szinte bármilyen elektronikus eszközzel, amely digitális jeleket használ a kommunikációra vagy bármely elektromos eszközzel, amellyel analóg jelekkel kommunikálni lehet, a micro:bit képes lehet kommunikálni. Hasonlóan megfelelő feszültségszinttel (3 V-os egyenárammal) működő motorokat is képes az eszközünk vezérelni, így akár egy valódi robot vezérlésére is képes lehet. Ha pedig egy micro:bit képességei kevesek lennének minderre, akkor több micro:bit együttműködve lehet erre képes (ne feledjük: a micro:bit-ek képesek egymással rádiójelekkel kommunikálni).

Mindebből látható, hogy bár a témakör végére értünk, a micro:bit lehetőségeinek még csak a felszínét ismertük meg. A webes fejlesztő felületén a kategóriákban még van néhány érdekes terület, így ha van rá lehetőség, akkor még bőven lehet az eszközzel kísérletezni – de már nem a Digitális kultúra tantárgy órái keretében.

ISBN: még nincs, de talán lesz...

Tankönyvként nem akkreditált oktatási segédanyag a NAT2020-hoz készült ke- rettanterv szerinti Blokkprogramozás és Robotika témakörök összevont tanítására a **Kiskunfélegyházi Móra Ferenc Gimnázium** hatévfolyamos gimnáziumi 7–8. évfo- lyama számára.

Mivel a dokumentum tartalma túlnyomó részt az irodalomjegyzék első pontjában szereplő dokumentum tartalmával egyezik meg, így a felhasználásához Dr. Abonyi- Tóth Andor (ELTE) írásbeli engedélye szükséges.

Felhasznált irodalom

- A dokumentumban felhasznált tartalmak nagyrésze Dr. Abonyi-Tóth Andor: Programozzunk micro:biteket! című segédanyagából származik (ELTE, Budapest, 2018.)

- Wikipedia: Robotics címszó, <https://en.wikipedia.org/wiki/Robotics>, 2020. május 28-i állapotban.
- Az elágazásoknál látható, sokfelé elágazó tábla a Magyar Kétfarkú Kutypárt Facebook oldalán található fényképből lett kivágva.
- [https://en.wikipedia.org/wiki/Sprite_\(folklore\)](https://en.wikipedia.org/wiki/Sprite_(folklore))
- Papírzongora: <https://www.facebook.com/groups/1194017457408416/permalink/1198450353631793/>
- Csak nyugalom: <https://projects.raspberrypi.org/en/projects/frustration>

