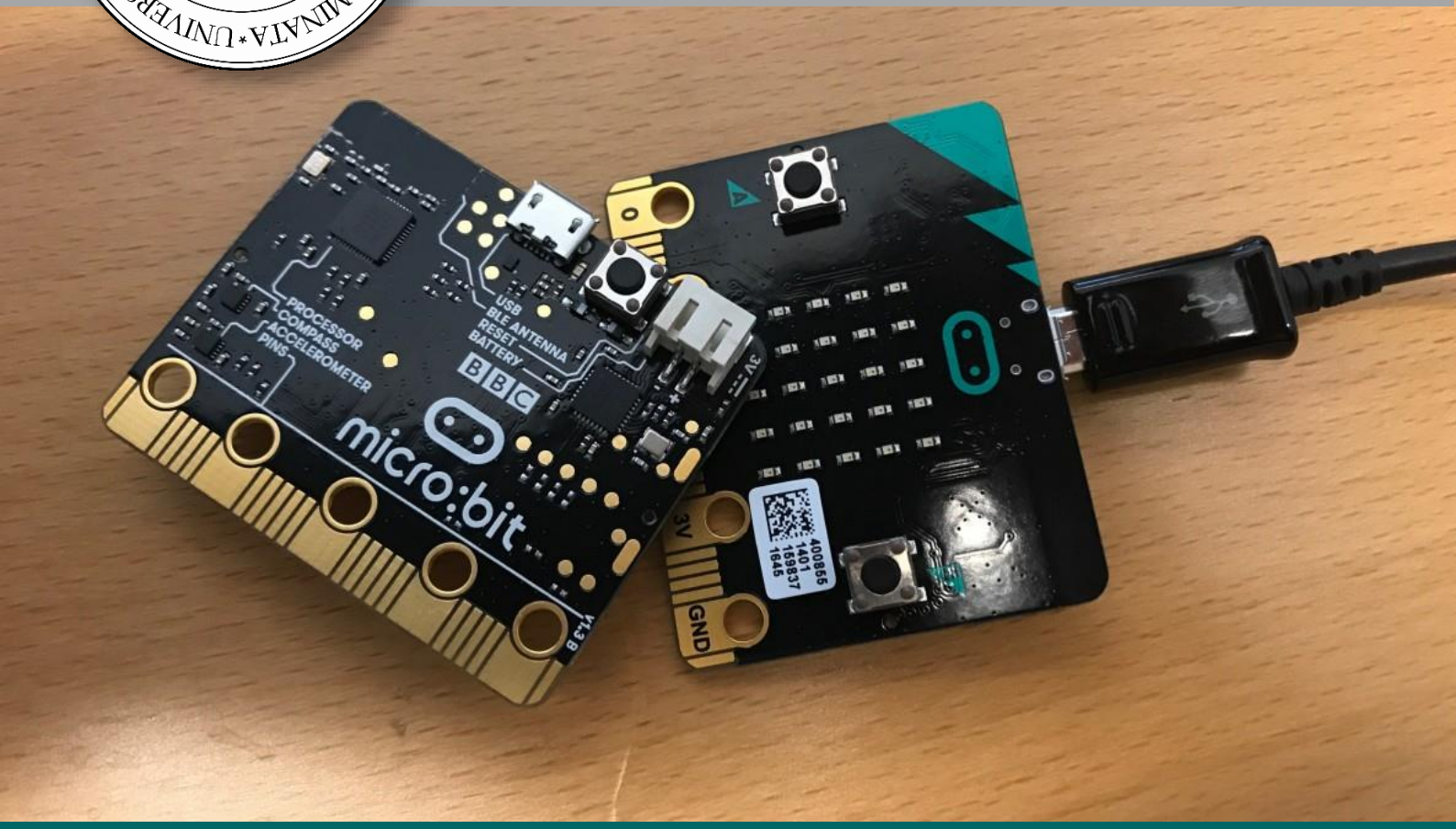
Belépő a tudás közösségébe

Szakköri segédanyag tanárok számára



Programozzunk micro:biteket!

Dr. Abonyi-Tóth Andor

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

Eötvös Loránd Tudományegyetem
Informatikai Kar

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

Programozzunk micro:biteket!

Szerző

Dr. Abonyi-Tóth Andor

Felelős kiadó

ELTE Informatikai Kar

1117 Budapest, Pázmány Péter sétány 1/C.

ISBN szám

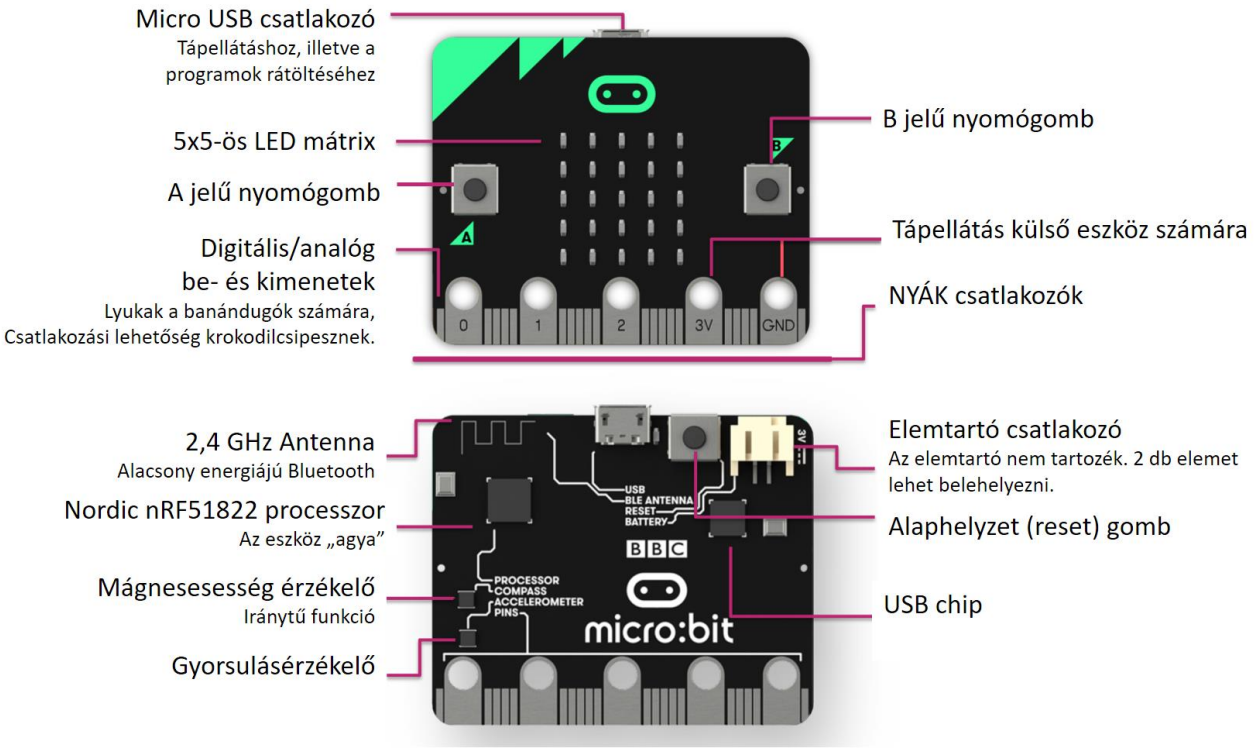
ISBN 978-963-284-992-8

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

Programozzunk micro:biteket!

A micro:bit bemutatása

A BBC micro:bit egy kifejezetten oktatási célra létrehozott, egylapkás mikrovezérlő, amely 4x5 cm-es méretével, 5x5-ös LED kijelzőjével, gyorsulásérzékelő, hőmérséklet érzékelő, fényérzékelő, irány érzékelő szenzoraival, be- és kimeneti csatlakozóival, 2 gombjával, bluetooth/rádió kapcsolódási lehetőségével igen sokrétű alkalmazást tesz lehetővé, legyen az (akár többfelhasználós) játék fejlesztése, viselhető eszközök (pl. okosóra, lépésszámláló, okosruha) tervezése és megvalósítása, kísérletezés a szenzorok által mért adatok felhasználásával, vagy éppen külső eszközök vezérlése/irányítása.



1. ábra A micro:bit felépítése

Az eszköz az angol BBC ötlete és koordinálása alapján került kifejlesztésre 29 partner bevonásával, annak érdekében, hogy a diákok minél korábban betekintést nyerjenek a programozás, illetve a mérnöki tudományok alapjaiba, ezzel is ösztönözve őket, hogy később a STEM területekkel kapcsolatos pályát válasszanak[1].

---

[1] Anthony, Sebastian: "BBC Micro:bit—a free single-board PC for every Year 7 kid in the UK (8 July 2015)

https://arstechnica.co.uk/gadgets/2015/07/bbc-microbit-a-free-single-board-pc-for-every-year-7-kid-in-the-uk/

Programozzunk micro:biteket!

A lapka 20 db NYÁK csatlakozó érintkezővel, 3 db digitális/analóg gyűrűs csatlakozóval és 3V-os kimeneti csatlakozóval rendelkezik, amelyek számos külső eszköz (szenzorok, szervó motorok, LED-ek, hangszórók, stb.) csatlakoztatására adnak lehetőséget, így az eszköz alkalmas arra is, hogy a robotika témakörébe bevezessük a diákokat, akár úgy is, hogy a megépített eszközöket a saját mobil eszközeik segítségével irányíthassák.

Az eszköz „csak" egy mikrovezérlő, ezért a felprogramozásához szükséges egy számítógép (asztali, notebook, vagy akár tablet és okostelefon), amellyel vagy USB kábellel, vagy Bluetooth kapcsolaton keresztül kapcsolódhatunk.

Megvásárolható kiszerelések

A micro:bit-eket többféle csomagban, kiszerelésben is meg lehet vásárolni.

BBC micro:bit alap panel (egyéni vásárlóknak ajánlott)

A csomag tartalma:

A csomag 1 db micro:bit lapkát tartalmaz, semmi mást, vagyis USB kábelt is külön kell vásárolni hozzá, ha azzal nem rendelkezünk.

BBC micro:bit go csomag (egyéni vásárlóknak ajánlott)

A csomag tartalma:

- 1 db BBC micro:bit panel
- 1 db elemtartó kábellel és csatlakozóval
- 2 db AAA elem
- 1 db USB táp/adat kábel programozáshoz
- 1 db rövid leírás

BBC micro:bit club csomag (iskolák számára ajánlott)

A csomag tartalma:

- 10 db BBC micro:bit panel
- 10 db elemtartó kábellel és csatlakozóval
- 20 db AAA elem
- 10 db USB táp/adat kábel programozáshoz
- 10 db rövid leírás

Programozzunk micro:biteket!

Ahhoz, hogy az eszközt megóvjuk, érdemes azokhoz tokokat is vásárolni. Tokokból is többféle áll rendelkezésre.

2. ábra MI:pro típusú tok

3. ábra A tok hátsó része, rögzített elemtartóval

4. ábra Plexi tok

5. ábra
Az elemek ennél a típusnál csak a tartó lecsavarása után érhetőek el.

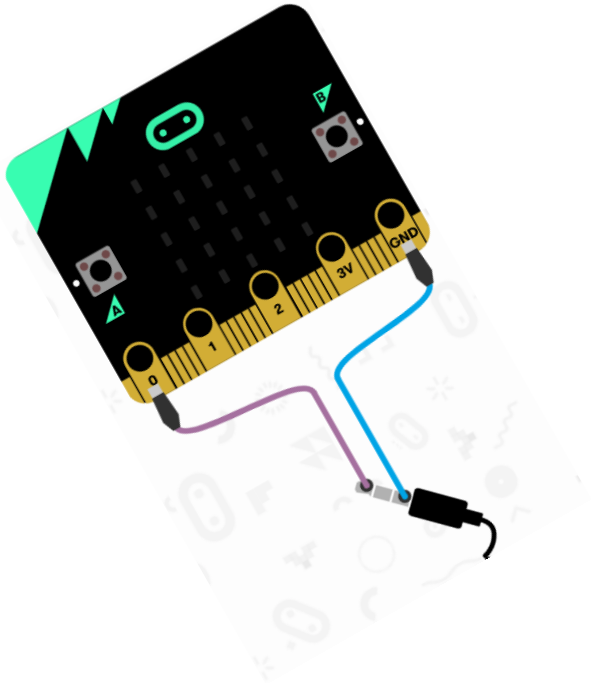
Fülhallgató, vagy hangszóró csatlakoztatása

A micro:bit ugyan nem tartalmaz beépített hangszórót, de képes arra, hogy a kivezetésein hozzá csatlakoztatott fülhallgatón/hangszórón hangokat, dallamokat játsszon le.

Vásárolhatunk olyan kábelt, amelybe a fülhallgató, vagy a hangszóró (úgynevezett *jack*) csatlakozóját tudjuk bele dugni. Ezen kábel csatlakozóit a 0 és GND (föld) csatlakozóhoz kell csíptetni.

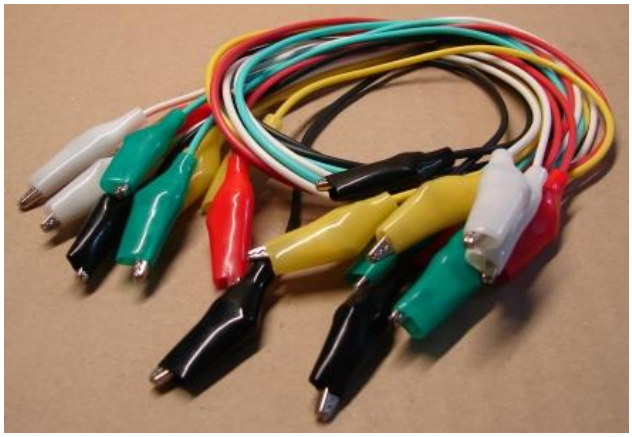
6. ábra Kábel fülhallgató/hangszóró csatlakoztatásához

Amennyiben nincs ilyen csatlakozónk, viszont vannak krokodilcsipeszeink, akkor a 0 és GND kivezetéseket közvetlenül a fülhallgató, vagy a hangszóró Jack dugójának megfelelő részeivel is összeköthetjük, mint ahogy azt a mellékelt ábrán is láthatjuk.



7. ábra A kábel csatlakoztatásának módja

Krokodilcsipeszek, banándugók



8. ábra Krokodilcsipeszek

9. ábra Banándugók

A krokodilcsipeszek, illetve banándugók segítségével a micro:biteket egymással, illetve külső eszközökkel is összekapcsolhatjuk. Ezen csipeszek felhasználásával igazán érdekes alkalmazásokat készíthetünk, amelyek közül párat az utolsó szakköri foglalkozás keretében be is mutatunk.

Az elérhető programozási környezetek bemutatása

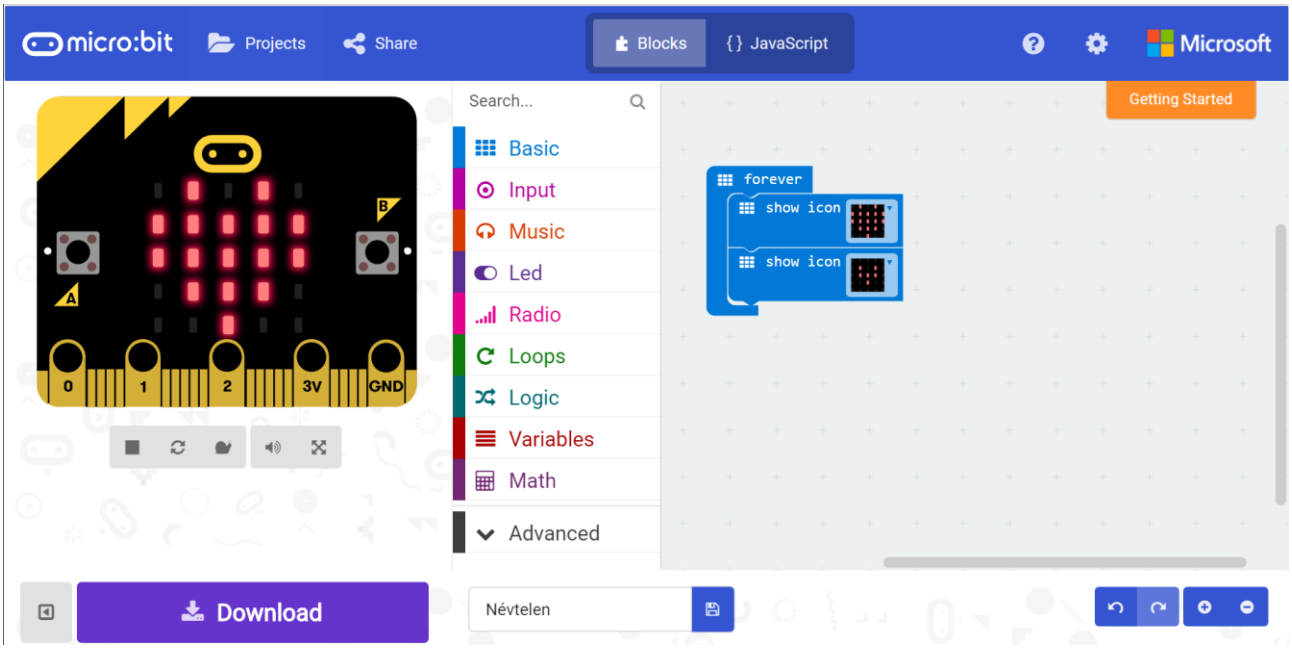
A micro:bit programozása asztali, illetve mobil környezetben is lehetséges, használhatunk vizuális blokknyelvet (JavaScript Blocks Editor[2]), de akár Python alapú programozási környezetben[3] is írhatjuk a kódot, amelyet USB, illetve Bluetooth kapcsolat segítségével tölthetünk fel a eszközre.

Ezen környezeteket a http://micro:bit.org/code/ webcímen érhetjük el.

JavaScript Blokk szerkesztő

A JavaScript blokk szerkesztő felület a https://makecode.micro:bit.org címen érhető el.

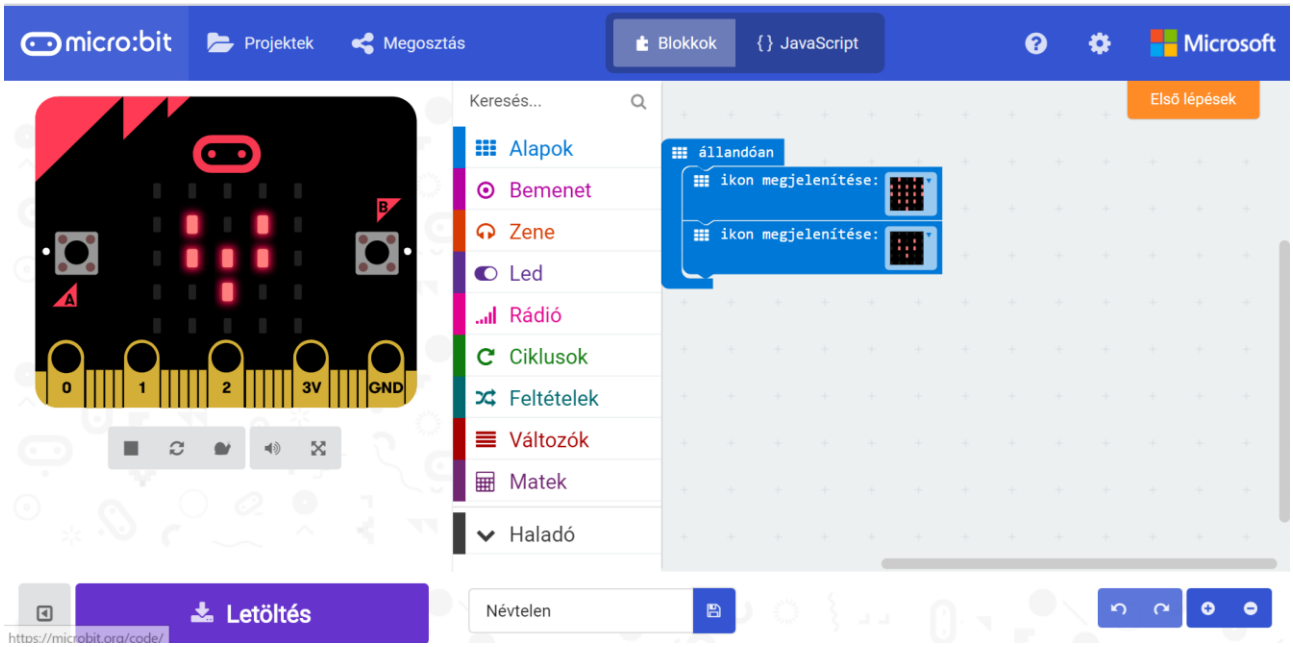
A felület ideális kezdők számára, hiszen blokkokból állíthatják össze az alkalmazásokat (mint pl. a Scratch nyelv esetén), a képernyő bal oldalán látható szimulátorban pedig ellenőrizni lehet a kód eredményét anélkül, hogy azt az eszközre rátöltenénk.



10. ábra Angol nyelvű felület

[2] https://makecode.microbit.org/

[3] http://python.microbit.org



11. ábra Magyar nyelvű felület

A két felület között úgy is válthatunk, hogy a webcímben paraméterben állítjuk be a kívánt nyelvet:

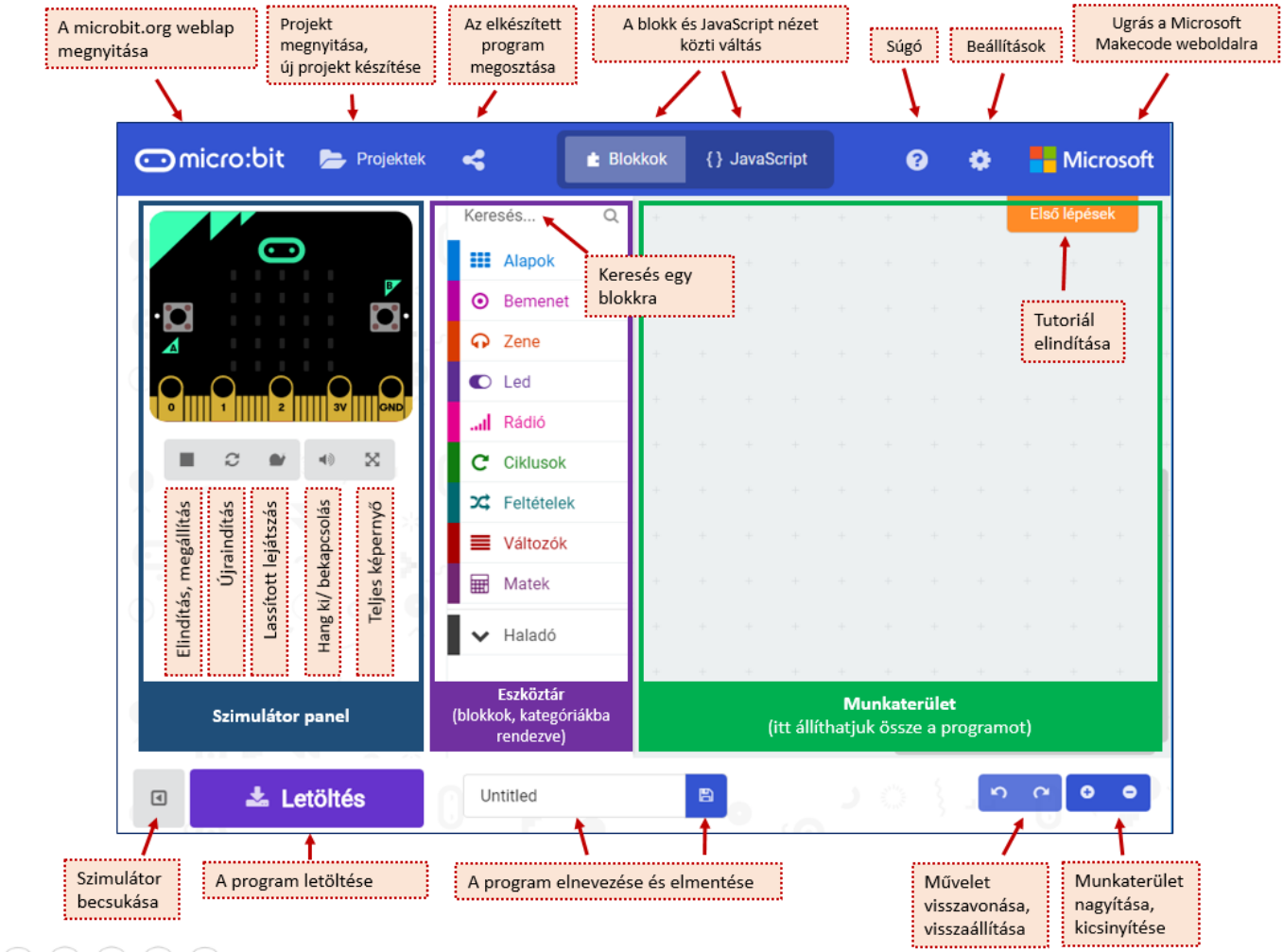
- https://makecode.microbit.org/?lang=hu
  ez a magyar felület hivatkozása
- https://makecode.microbit.org/?lang=en
  ezzel pedig az angol felületet jeleníthetjük meg

A kódolással most ismerkedő, kisebb gyermekek tanítása során a magyar felület használatát javasoljuk, azonban egy idő után itt is érdemes lehet áttérni az angol felületre, mivel az angol parancsok (pl. *if-then-else, repeat, for*, stb.) megismerése után könnyebb lesz áttérni más programozási nyelvekre.

Kiadványunkban a példakódoknál és a leírásoknál az angol kifejezéseket használjuk.

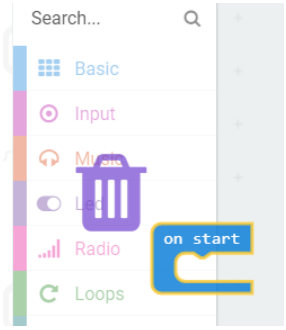
A felület rövid bemutatása

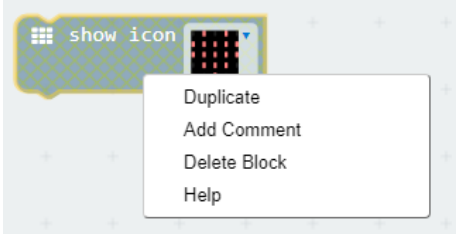
A programozási felület az alábbi főbb részekből áll (A képet nagyobb változatban a mellékletben is elhelyeztük, mind a magyar, mind az angol felületre vonatkozóan):



Az eszköztáron található **blokkokat** fogd és vidd módszerrel **vonszolhatjuk a munkaterületre**.

A blokkot úgy **törölhetjük**, hogy visszavonszoljuk az eszköztár területre, ekkor egy kuka ikon megjelenik a terület felett. A blokkot a billentyűzet *Del* gombjával is törölhetjük, ha előtte rákattintottunk az egérrel.





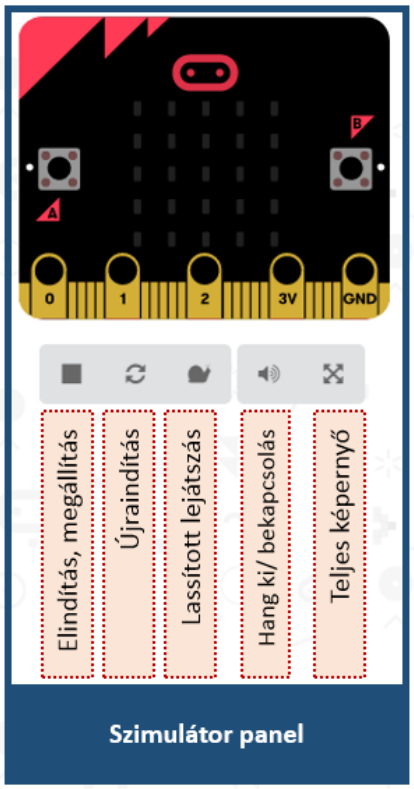
A blokkra a **jobb egérgombbal** kattintva egy helyi menü jelenik meg. Itt szintén megtaláljuk a törlési lehetőséget (*Delete Block* / *Blokk törlése*). Emellett lehetőségünk van arra, hogy az adott blokkból **másolatot**, duplikátumot készítsünk (*Duplicate* / *Másolat készítése*), **magyarázattal** lássuk el (*Add comment* / *Megjegyzés hozzáadása*), vagy **segítséget** kérjünk a használatára vonatkozóan (*Help* / *Súgó*).

Programozzunk micro:biteket!

A blokk alapú programozás előnye, hogy csak olyan blokkokat illeszthetünk össze, amelyek egymáshoz illenek, vagyis szintaktikailag helyes kód lesz az eredmény. Így a program az esetek többségében le fog futni, akkor is, ha nem az elvárt, kigondolt eredményt adja. Természetesen lehetnek olyan esetek is, amikor a program nem futtatható, ha például egy számot akarunk kiíratni, a változó azonban szöveget tartalmaz szám helyett. Ekkor hibaüzenetet fogunk kapni, amely magyarul így hangzik: „*Sajnos nem tudtuk lefuttatni ezt a projektet. Kérlek keresd meg a kódodban a hibát!*”

12. ábra Hibaüzenet az angol felületen

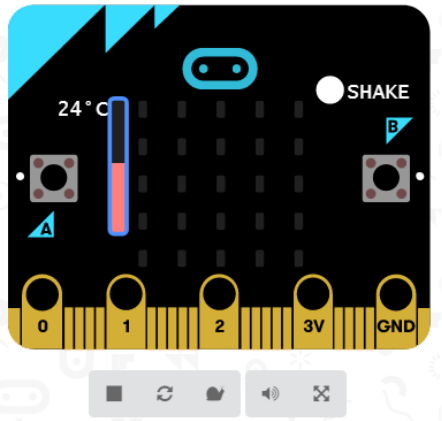
A szimulátor



A bal oldalon található szimulátor segítségével a kódot anélkül is futtathatjuk, hogy a micro:bit eszközre kellene töltenünk azt.

Érdemes tesztelni a szimulátorban a munkánkat, és az esetleges hibákat javítani, mielőtt az eszközre töltenénk, mert így gyorsabban, hatékonyabban dolgozhatunk.

A szimulátorban nem csak a gombnyomásokat szimulálhatjuk, hanem az egyes gesztusokat is. Ha például a kódunkban a *Rázás* (*shake*) eseményt használjuk, a szimulátorban is megjelenik egy rázás (shake) gomb. Ha pedig a szenzorokat használjuk, azok értékét is tudjuk módosítani a szimulátorban (pl. az iránytű esetén be tudjuk állítani az irányt, a fényérzékelő esetén a fényerősségszintet, hőmérséklet mérés esetén az adott hőmérsékletet, és így tovább)



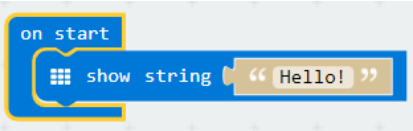
Ezen a képen láthatjuk, hogy a *Rázás* (shake) eseményt is tudjuk szimulálni, illetve a hőmérsékleti értéket is be tudjuk állítani. Ezen lehetőségek viszont csak akkor jelennek meg a felületet, ha azokat ténylegesen használjuk a kódunkban.

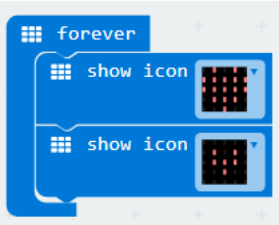
A blokkok fajtái

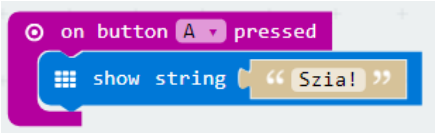
A felhasználható blokkoknak több fajtája is létezik. Vannak olyan blokkok, amelyek a program főbb **vezérlő szerkezeteit** jelentik. Ilyen például az `on start` (indításkor), a `forever` (állandóan), vagy az `on button A pressed` (amikor a(z) „A” gomb lenyomva).

Mi a különbség ezek között?

- Az `on start` blokk tartalma csak egyszer fut le, a program elindulásakor.
- A `forever` blokk tartalma a háttérben folyamatosan, ciklikusan ismétlődve fut le. Ha például folyamatosan ismétlődő animációt akarunk készíteni, ez a blokk hozzá a megfelelő.
- Az `input` kategóriában több eseményt is találunk. Az `on button A pressed` blokk tartalma akkor hajtódik végre, ha megnyomtuk az „A” gombot.

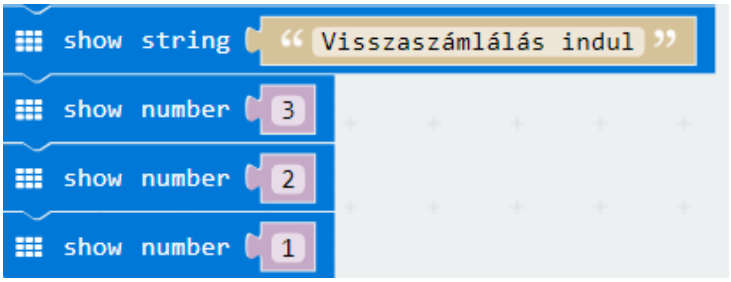




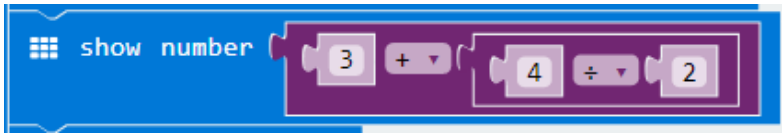


Látszik, hogy ezen blokkoknak nincs külső kapcsolódási lehetőségük, csak ezeken belül tudunk más blokkokat elhelyezni.

Egy másik csoportot alkotnak azon blokkok, amelyeket **egymás után tudunk elhelyezni**. Ezek felső részén bemélyedést látunk, az alsó részén pedig kitüremkedést, jelezve, hogy ezen blokkokat egymás alatt lehet elhelyezni, ezek egymás után hajtódnak végre, mint ahogy ez az alábbi példában is látható.



Ezen kívül vannak olyan blokkok is, amelyek oldalról illeszkednek a blokkokhoz, akár úgy is, hogy azokat többszörösen egymásba is ágyazhatjuk. Ezek segítségével átadhatunk paramétereket, megfogalmazhatunk logikai kifejezéseket, kiszámolhatunk matematikai kifejezéseket, és így tovább.

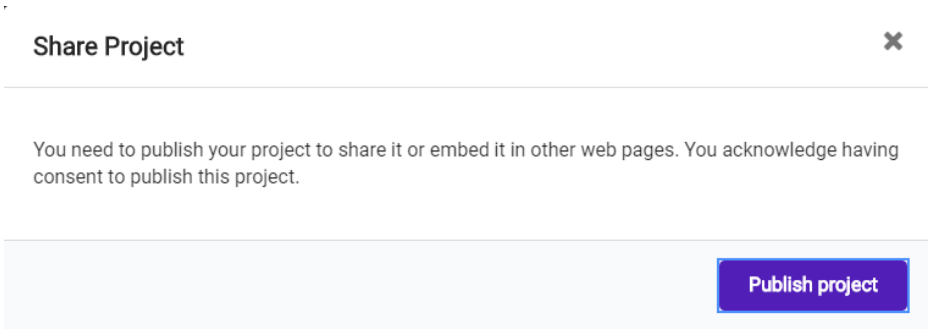


A munkák megosztása

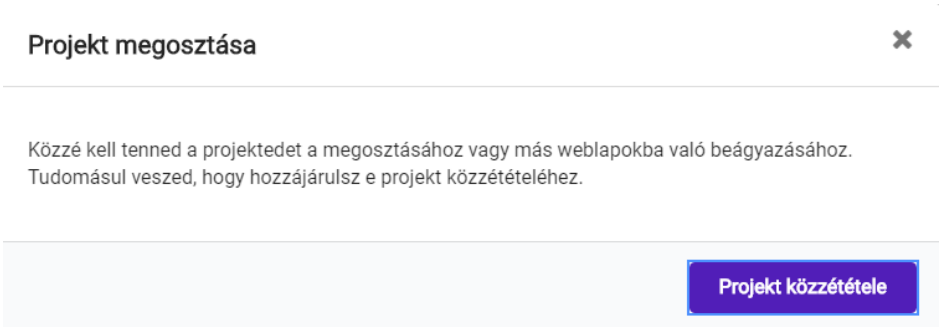
Az elkészített munkákat nagyon egyszerűen megoszthatjuk egymással. Ehhez a 

(Megosztás) gombra kell kattintanunk.

Ekkor megjelenik az alábbi ablak:



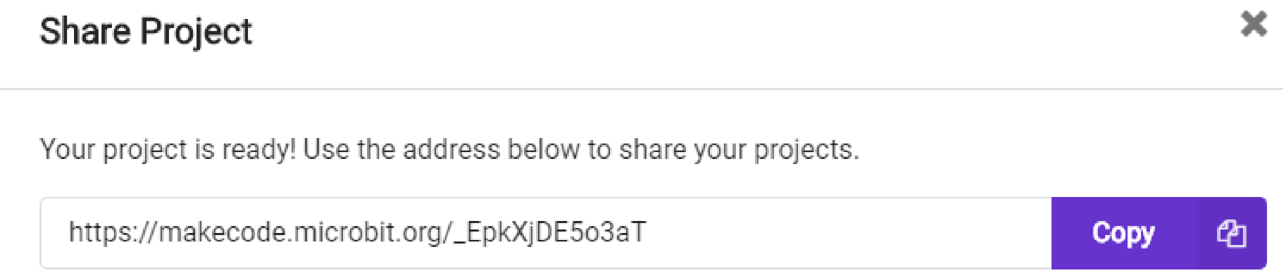
13. ábra Angolnyelvű felület



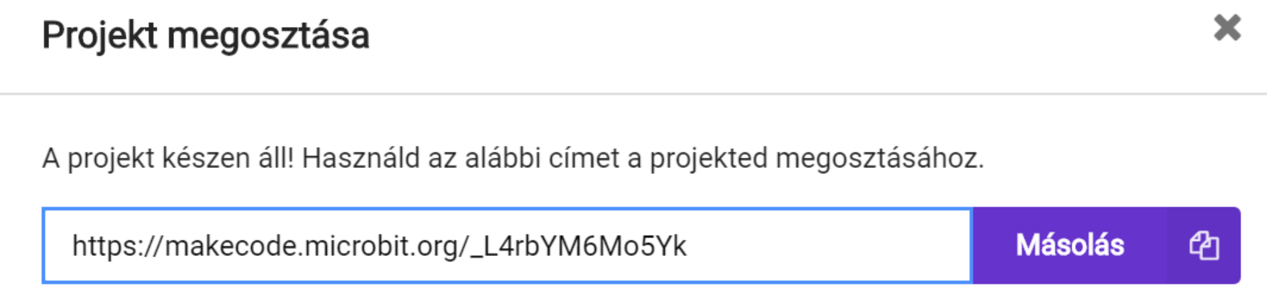
14. ábra Magyar nyelvű felület

Itt kattintsunk a *Publish project* / *Projekt közzététele* gombra.

Ezután megkapjuk a projektünk webcímét, amelyet a *Copy* / *Másolás* gombbal a vágólapra másolhatunk. Ezt a linket tetszőleges helyen megoszthatjuk (közösségi oldal, email, stb.).



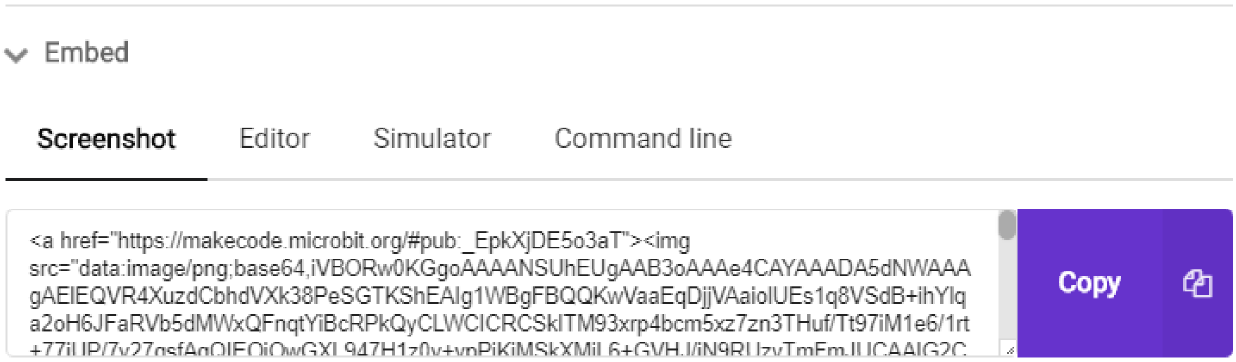
15. ábra Angol nyelvű felület



16. ábra Magyar nyelvű felület

Beágyazás

A munkákat nem csak megoszthatjuk, hanem akár be is ágyazhatjuk weboldalakra. Ehhez az *Embed* / *Beágyazás* szakaszban a megfelelő fülecskékre kattintva kapjuk meg a beágyazó kódot, attól függően, hogy képernyőképet, a szerkesztő felületet, a szimulátort szeretnénk beilleszteni. Az utolsó lehetőség (Command line/ Parancssor) segítségével olyan kódot kapunk, amelyet a felület parancssori változatában használhatnánk fel.



17. ábra Angol nyelvű felirat

18. ábra Magyar nyelvű felirat

Program letöltése és az eszközre másolása

Amikor elkészültünk a projektünkkel, akkor a Letöltés gombra kattintva egy *hex* kiterjesztésű állomány töltődik le a gépünkre. A micro:bitet USB kábellel csatlakoztassuk a számítógépünkhöz. Ekkor egy meghajtóként fogjuk elérni az eszközt (pl. E meghajtó), mintha csak egy pendrive lenne. A letöltött hex állományt másoljuk rá az eszközre. Ennek során az eszközön lévő sárga LED villogni fog. Amennyiben a másolás megtörtént, a villogás megszűnik. Ezek után már kipróbálható a program.

Webinárium a felület használatáról

A felülettel való megismerkedéshez a https://youtu.be/f9EHG8dnSnM címen elérhető videót is ajánljuk. A webináriumban (amely a Kódolás hete 2017 rendezvényhez kapcsolódik) két előadást is láthatunk. Először Abonyi-Tóth Andor (ELTE IK) mutatja be a micro:bit eszköz főbb tulajdonságait, és azt, hogy hogyan készíthetünk egyszerű játékokat a gyerekekkel (50 perc). Ezt követően Nádori Gergely (Alternatív Közgazdasági Gimnázium) az eszköz külső csatlakozói által elérhető funkciókat mutatja be.

A szakkör felépítése

Alkalmak száma	14
Óraszám/alkalom	2 * 45 perc = 90 perc
Ajánlott korosztály	10 – 16 év
Eszközszükséglet 15 diákra + 1 tanárra vetítve	– 15+1 db asztali számítógép vagy notebook, egérrel – 15+1 db micro:bit (lehetőleg tokkal, elemtartóval) – 15+1 db USB kábel a számítógép és a micro:bit összekötéséhez (USB A – micro USB) – 15+1 db hangkábel, vagy kétszer ennyi krokodilcsipesszel ellátott vezeték (ennek hiányában csak a szimulátorban lesz hallható a hang) – 1 db projektor – Előnyös, ha rendelkezésre áll egy LMS rendszer (vagy facebook csoport, google csoport) ahova a gyerekek fel tudják tölteni az általuk készített alkalmazások webcímét, hogy minden résztvevő láthassa a társak által elkészített munkákat, azokhoz hozzászólhasson.

Szakköri anyagunkat a kódolással ismerkedő diákok számára hoztuk létre, ezért a Makecode Blokk szerkesztőt használjuk, a példákat ezen környezetben hoztuk létre és osztottuk meg.

A mellékletben a felületen elérhető, gyakran használt blokkok rövid leírását közöljük. A dokumentum ezen részét nyomtatásban is érdemes kiadni a diákok számára. A foglalkozások során azonban nem minden blokkot érintünk az itt felsoroltak közül.

Leírásunk informatika tanárok számára készült, ezért nem térünk ki az alapvető programozással kapcsolatos fogalmak ismertetésére (pl. ciklus, elágazás). Kérjük a kollégákat, hogy ezen fogalmakat a megfelelő helyen, a szakköri anyagban található példák segítségével magyarázzák el a diákoknak. A példákat úgy igyekeztünk felépíteni, hogy egyszerre ne kelljen több fogalmat is bevezetni.

Kiadványunk csak egy lehetséges tematikát mutat be, ettől szabadon el lehet térni, sőt arra bíztatunk mindenkit, hogy módosítsa az anyagot, vigye bele saját ötleteit a foglalkozásokba. Ha úgy érezzük, hogy egy témakört más időkeretben szeretnénk tárgyalni, mert túl könnyű az adott célcsoportnak, vagy ellenkezőleg, nehezebben volt érthető a diákok számára, vagy

sok olyan fejlesztési ötlet merült fel, amelyeket érdemes megvalósítani, változtassuk meg nyugodtan az időkereteket.

Célunk az volt, hogy mind az animációk, mind a játékfejlesztés, mind a szenzorok használata és a barkácsolás témaköre is megjelenjen a szakkör során, de ne igényeljen olyan extra eszközöket, amelyek nem feltétlenül állnak rendelkezésre az iskolákban (pl. külső szenzorok, vonalkövető járművek, robotok, stb.) Amennyiben ilyen eszközök rendelkezésre állnak, feltétlenül mutassuk meg a gyerekeknek.

Az anyagot időnként szeretnénk bővíteni, a visszajelzések alapján módosítani, alternatív megoldásokkal, vagy akár konkrét diák munkákkal bővíteni, ezért minden visszajelzést szívesen fogadunk az https://www.facebook.com/tethalo/ facebook oldalunkon.

A kiadványunkban bemutatott példák, alkalmazások mellett a https://www.micro:bit.co.uk/blocks/lessons oldalon található példák tanulmányozását, felhasználását, továbbfejlesztését is ajánljuk.

Ezen kívül minden kolléga figyelmébe ajánljuk a következő facebook csoportokat:

- micro:bit tanári csoport (magyar)
  https://www.facebook.com/groups/898764273601915/
- micro:bit műhely (magyar, elsősorban diákok számára)
  https://www.facebook.com/groups/1194017457408416/
- BBC Microbit Computer (angol)
  https://www.facebook.com/groups/1756471244599979/

A témák

Szakkörünkben az alábbi témaköröket érintjük:

1. Rajzok/animációk megjelenítése a LED mátrixon
2. Rajzok/animációk megjelenítése a LED mátrixon, zenével
3. Animáció több eszközön – csoportmunka
4. Egymásba ágyazott ciklusok, elágazások használata, változó értékének növelése, csökkentése
5. Labirintus játék: while ciklus, logika feltételek, tagadás, függvények használata
6. Haladó játékfejlesztés spriteokkal (1. rész)
7. Haladó játékfejlesztés spriteokkal (2. rész)
8. Saját játékok megvalósítása
9. Ismerkedés a szenzorokkal
10. Játék a szenzorokkal
11. Kommunikáció a micro:bitek között
12. Többfelhasználós játékok - Saját játékötletek megvalósítása (1. rész)

Programozzunk micro:biteket!

13. Többfelhasználós játékok - Saját játékötletek megvalósítása (2. rész)
14. Barkácsoljunk együtt!

Felépítés, jelölések

Minden szakköri alkalom egy táblázattal kezdődik, amelyben láthatjuk az adott alkalom időbeosztását. Ez után kerül kifejtésre az adott tanári, vagy tanulói tevékenység.

A tanár által bemutatandó, a diákokkal közösen elkészítendő projektek esetén minden esetben szerepeltetjük az anyagban az elkészült program elérhetőségét, valamint azon lépéseket, amelyek követésével az alkalmazás elkészíthető.

A diákok által elkészítendő feladatokat más színnel (világos zöld fejléc) jelöljük. Ezeket a foglalkozás során ki is vetíthetjük. Magát a szakköri anyagot nem feltétlenül érdemes megosztani a diákokkal, mivel több, csak a tanárok számára szóló instrukciót is tartalmaz.

A diákok által megvalósítandó projektek esetén, amennyiben van mintamegoldás, az a villanykörte ikonra kattintva elérhető.

A programozási **blokkok kategóriáit** más színnel jelöljük, mint a **blokkok elnevezését**.

1. alkalom (Rajzok/animációk megjelenítése a LED mátrixon)

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Az eszköz bemutatása, munkavédelmi előírások	Frontális tanári bemutató	8 perc
2. Szívdobbanás animáció	Tanári bemutató, új funkciók megismerése (szöveg kiírása, ikon megjelenítése, várakozás)	7 perc
3. Szívdobbanás animáció variációja	A diákok egyénileg továbbfejlesztik az előbbi alkalmazást, megosztják, és megnézik egymás munkáit. A tanár körbejár, válaszol a felmerült kérdésekre, a legjobb műveket frontálisan is bemutatja.	10 perc
4. Integető robot	Tanári bemutató, új funkciók megismerése (saját ikon rajzolása, animáció készítés, gombnyomás kezelése, ismétlés valahány alkalommal, rázás)	5 perc
5. Integető robot - variáció	A diákok egyénileg továbbfejlesztik az előbbi alkalmazást, megosztják, és megnézik egymás munkáit. A tanár körbejár, válaszol a felmerült kérdésekre. A diákok bemutatják egymásnak a munkáikat.	15 perc
6. Nyilak megjelenítése az eszköz döntésekor	Egyéni feladat a diákok számára. A tanár körbejár, válaszol a felmerült kérdésekre.	10 perc
7. Szabadon választható feladatok, egyéni animációk készítése	A diákok saját ötlet alapján készíthetnek animációkat, vagy választhatnak a kiadott feladatok közül. A tanár körbejár, válaszol a felmerült kérdésekre.	25 perc
8. Az elkészült munkák megtekintése.	A diákok bemutatják egymásnak az elkészült munkákat. Közösen megbeszéljük az egyes munkák pozitívumait, és a lehetséges továbbfejlesztési ötleteket.	10 perc

1. Munkavédelmi előírások, az eszköz bemutatása

Munkavédelmi előírások

Mielőtt a gyerekek kezébe adjuk az eszközöket, hívjuk fel a figyelmüket pár fontos tudnivalóra!

- Figyeljünk arra, hogy a szakköri tevékenységek során se az eszközökben, se a társainkban ne tegyünk kárt!
- A munkakörnyezetet úgy alakítsuk ki, hogy ne eshessen le, illetve véletlenül se lehessen lesodorni a micro:bitet!
- Mindig tiszta felületen dolgozzunk, ahol nincsenek olyan tárgyak, amelyek rövidzárlatot okozhatnak (tűzőgép kapocs, gemkapocs)!
- Használjuk az USB kábelt! Elemről csak a szükséges ideig működtessük az eszközt!
- Ne használjunk Li-Ion vagy Li-polymer cellákat, akkor sem, ha hasonló csatlakozókkal is szerelték őket, mert azok a termék tönkremeneteléhez vezethetnek!
- Mielőtt az eszközhöz hozzáérnének, érintsük meg a számítógép házát, vagy a tanteremben lévő radiátor, vagy fűtési csőhálózat felületét, mert így védekezhetünk az elektrosztatikus kisülés ellen, amely akár tönkre is teheti az eszközt.
- A különböző gesztusok kipróbálásakor (pl. rázás, eldöntés stb.) biztosan tartsuk az eszközt a kezünkben, nehogy véletlenül leejtsük.
- Az eszköz érzékelőit az erős mágneses tér, fém tárgyak, stb. megzavarhatják. Ha a szenzorok „furcsa” értékeket mérnek, győződjünk meg arról, hogy a közelben nincs-e zavaró tényező.7
- Ne kössünk a micro:bithez nem kompatibilis eszközöket! Ezek lehet, hogy nagyobb tápfeszültséget igényelnek, mint amit az eszköz biztosítani tud. Emiatt túlmelegedhet az eszköz, vagy akár tönkre is mehet! Rosszabb esetben égési sérüléseket is okozhat!

Az eszköz bemutatása

Ezután röviden, élő szóban mutassuk be az eszközt a diákoknak.

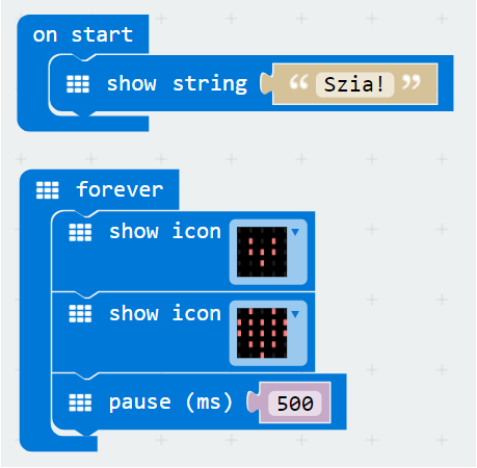
Térjünk ki arra, hogy

- milyen gombok, csatlakozók, érzékelők találhatóak rajta,
- hogyan lehet a számítógéphez csatlakoztatni az USB kábel segítségével
- ha van elemtartó, akkor a csatlakozóját hogyan lehet a panelhez csatlakoztatni, illetve róla eltávolítani

2. Szívdobbanás animáció készítése

A micro:bit egy 5x5-ös LED mátrix kijelzőt is tartalmaz, amelyen (gördülő) szövegeket, számokat, ikonokat, sőt animációkat is megjeleníthetünk. Kezdjük az eszköz programozását ezzel!

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_cqCWCFgKug1X
Készítsünk egy egyszerű animációt, amely azzal indul, hogy kiírjuk a „Szia!” szöveget, majd egy szív nagyobb és kisebb változatát váltogatjuk, ezzel elérve, mintha dobó szívet látnánk. Az egyszerű alkalmazás elkészítése során mutassuk be magát a programozási környezet is: • Hogyan tudjuk az egyes blokkokat kiválasztani, a felületre húzni? • Hogyan törölhetjük a blokkokat? • Hogyan használhatjuk a szimulációs lehetőséget? • Hogyan menthetjük el a projektet és tölthetjük az eszközre? • Hogyan készíthetünk új projektet, hogyan importálhatjuk a már elmentett kódunkat? • Hogyan oszthatjuk meg az általunk készített programot? • Hogyan válthatunk át a JavaScript nézetre, illetve onnan a Blokk nézetre?



- az `on start` blokk tartalma a futás kezdetén hajtódik végre. Helyezzük el ebben a **`Basic / Show string`** blokkot, és írjuk át a „Hello!” szöveget arra, hogy „Szia!”
- Ahhoz, hogy folyamatosan ismétlődjön a szívdobogás animáció, a `forever` blokkot kell felhasználnunk. Ennek tartalma gyakorlatilag folyamatosan, végtelen ciklusban végrehajtódik.
- Ebben helyezzük el a **`Basic / show icon`** blokkot, és válasszuk ki itt a kisebb szívet.
- Duplikáljuk a blokkot a jobb egérgomb lenyomásával elérhető menüben, helyezzük a korábbi blokk alá, és válasszuk ki a nagyobb szív ikont.
- A szimulátorban ellenőrizzük az eredményt!

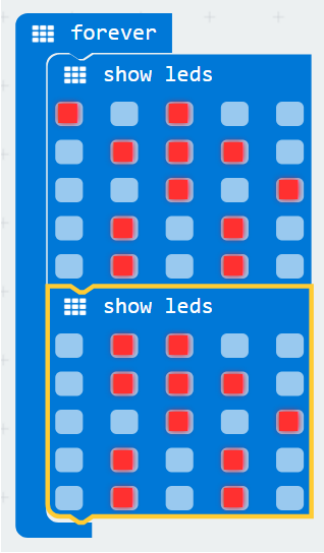
	– Hogy a nagyobb szív tovább látszódjon a képernyőn, mint a kisebb, a **Basic / pause (ms)** blokkot helyezzük el utána, és adjuk meg az 500 ms értéket, ami fél másodpercet jelent. o Hívjuk fel a figyelmet arra, hogy az animációkat csak azáltal is megváltoztathatjuk, hogy az egyes fázisok megjelenítési idejét módosítjuk. pl. folyamatos járásból tudunk bicegő járást készíteni, anélkül, hogy a fázisokat átrajzolnánk. – A diákok készítsék el ugyanezt a projektet, töltsék rá az eszközükre, majd próbálják ki.

3. Szívdobbanás variációja

Feladat a diákok számára
– Nem csak szöveget, hanem számot is meg tud jeleníteni az eszköz. Ehhez a **Basic / show number** blokkot kell felhasználni. A „Szia!” szöveg helyett jelenítsd meg a 3, 2, 1 számsort, vagyis számoljon visszafele az eszköz 3-tól 1-ig. – A szívdobogás helyett, a rendelkezésre álló ikonok felhasználásával készíts egy másik animációt! – Az elkészített animáció webcímét oszd meg a közös felületen! Nézd meg, hogy a társaid milyen animációkat készítettek!

4. Integető robot

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_1u6ieoJJfePE
A LED mátrix felett a micro:bit logója látható , amely olyan, mintha egy robot feje lenne. Használjuk ki ezt a lehetőséget, így csak a robot testét kell megrajzolnunk. Készítsünk olyan animációt, amelyben a robot folyamatosan integet a jobb kezével. Ekkor már nem előre megadott ikonokat használunk, hanem mi adjuk meg, hogy a LED mely pontjai legyenek kigyújtva.



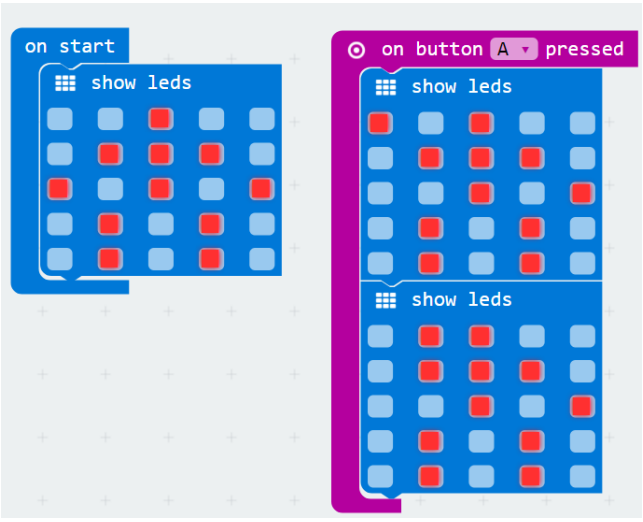
- Ahhoz, hogy folyamatosan ismétlődjön az animáció, a **forever** blokkot kell használnunk.
- Ebben helyezzük el a **Basic / show leds** blokkot, amelyben kattintsunk azokra a pontokra, amelyek be akarunk kapcsolni. Ezek fognak pirosan világítani.
- Duplikáljuk a blokkot, helyezzük a korábbi blokk alá, és módosítsuk az ábrát, hogy ezek lejátszásával integetés hatást érhessünk el.
- A diákok készítsék el ugyanezt a projektet, töltsék rá az eszközükre, majd próbálják ki.

A tanár által elkészítendő/bemutatandó mintaprojekt

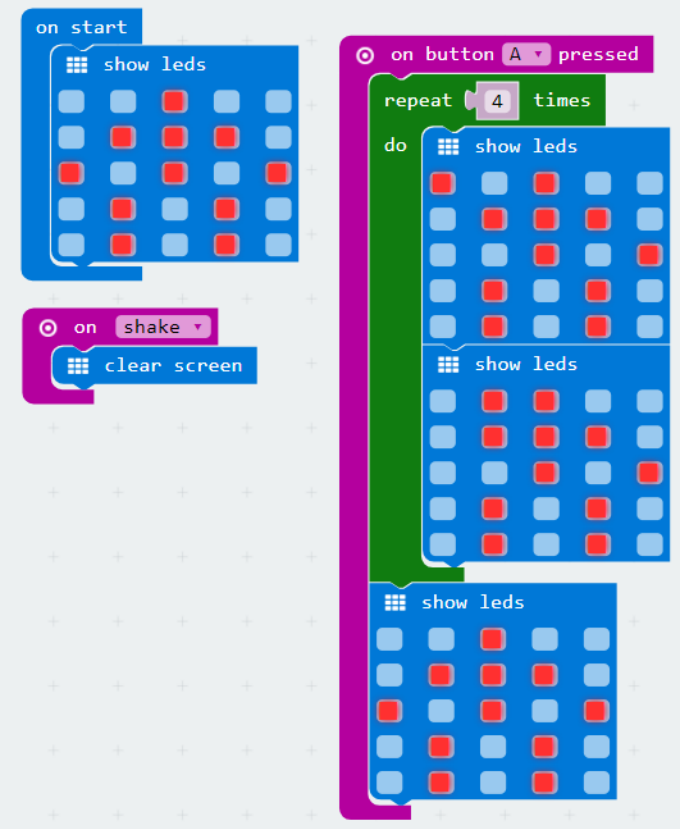
https://makecode.microbit.org/_K7H2acPvCWy3

Fejlesszük tovább a projektet úgy, hogy

- indításkor egy álló alak látszódjon,
- az „A” gomb hatására integessen a robot először egyszer, majd oldjuk meg, hogy négyszer ismétlődjön az integetés.
- Rázás hatására törlődjön le a képernyő.



- Az **on start** blokkban helyezzük el az álló alak megjelenítéséért felelős **show leds** blokkot.
- Az **Input / on button A pressed** blokk tartalma akkor hajtódik végre, ha megnyomtuk az „A” gombot. Ebbe a blokkba mozgassuk át a korábbi animációnkat a **forever** blokkból.
- Hívjuk fel a figyelmet arra, hogy a legördülő menüből választható ki a „B” és „A+B” gomb együttes lenyomása is.
- A diákok is készítsék el ugyanezt, és a szimulátorban próbáljuk ki a projektet!



- Jobb lenne, ha többször, pl. négyszer integetne a robot, majd visszaállna az eredeti, álló alak. Húzzuk be a **Loops / repeat x times do** blokkot, és magyarázzuk el ennek jelentését (a tartalom a megadott számú alkalommal hajtódik végre)
- Ha megrázzuk az eszközt, akkor legyen letörölve a képernyő. Ehhez az **Input / on shake** blokkot kell használnunk, amelyben el kell helyezni a **Basic/ clear screen** blokkot.
- A diákok készítsék el ugyanezt a projektet, mentsék el és töltsék rá az eszközükre.

5. Integető robot - variáció

Feladat a diákok számára

- Fejleszd tovább az alkalmazást úgy, hogy a „B” gomb megnyomásakor a robot a bal karjával integessen, szintén négyszer, majd jelenjen meg az álló alak!
- Ha az "A" és "B" gombot egyszerre nyomjuk meg, akkor a robot mindkét kezét emelje fel, majd engedje le!
- Akkor is integessen a robot, ha balra, vagy jobbra döntjük az eszközt! Ehhez az **Input / on shake** blokk legördülő menüjében a **tilt left**, **tilt right** pontokat kell kiválasztani.
- Találj ki egyedi robot alakot, vagy animációt arra az esetre, ha az eszközt felfelé, illetve lefelé billentjük. (Ez az esemény a **logo up**, illetve **logo down**. Ez arra utal, hogy a  logó a vízszintes helyzethez képest feljebb, vagy lejjebb kerül)

6. Nyilak megjelenítése az eszköz döntésekor

Feladat a diákok számára

– Készítsd egy új projektet.
– Ha az eszközt
  ○ balra döntjük (`tilt left`), akkor jelenjen meg egy balra mutató nyíl (Nyugat – West)
  ○ jobbra döntjük (`tilt right`), akkor jelenjen meg egy jobbra mutató nyíl (Kelet – East)
  ○ felfelé döntjük (`logo up`), akkor jelenjen meg egy felfelé mutató nyíl (Észak – North)
  ○ lefelé döntjük (`logo down`), akkor jelenjen meg egy lefelé mutató nyíl (Dél – South)
  ○ megrázzuk (`shake`) legyen letörölve a képernyő.
– Ehhez a korábban tanultak mellett fel kell használnod a **`Basic / Show arrow`** blokkját.
– Azt is meg tudjuk különböztetni, hogy a ledes kijelző felfelé (mennyezet felé), vagy lefelé (padló felé) néz. Ehhez a `screen up, screen down` eseményeket kell használnod. Fejleszd tovább a projektet úgy, hogy ha lefelé néz a kijelző, akkor legyen kigyújtva a középső pontja, ha felfelé, akkor pedig a sarkokban lévő 4 pont.

7. Szabadon választható feladatok, egyéni animációk készítése

További (szabadon választható) feladatok a diákok számára
Most már tudod, hogy hogyan írhatsz 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 megnyomásakor, 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. 5 másodperc elteltével zuhanjon le az egyik csillag, 2 másodperc múlva egy másik csillag, végül 2 másodperc múlva az utolsó csillag is.

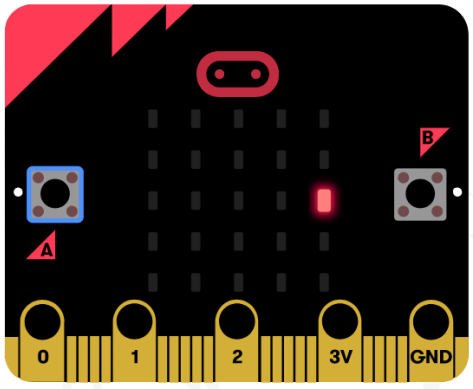
4. Készíts olyan animációt, amelyben a lehulló esőcseppekből egy tócsa alakul ki.
5. Készíts egy időjárás előrejelző alkalmazást. Az „A” gomb hatására jelenjen meg egy napsugár, a „B” gomb hatására egy esőfelhő. A két gomb együttes megnyomásakor egy hópihe alak látszódjon. Rázáskor legyen letörölve a kijelző.

2. alkalom (Rajzok/animációk megjelenítése a LED mátrixon, zenével)

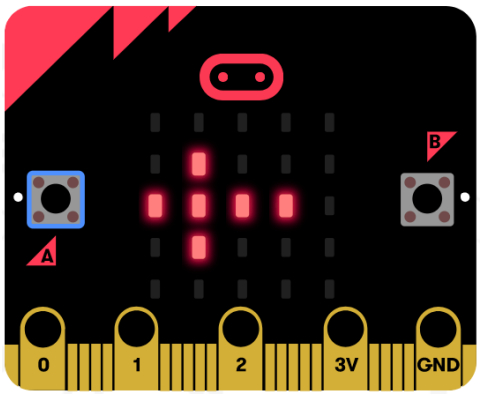
Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Ismétlés (nyílvessző animáció)	Ismétlő, ráhangolódó feladat A diákok egyénileg egy egyszerű, ismétlő feladatot oldanak meg.	5 perc
2. Animálás eltolással	Tanári bemutató, új funkciók megismerése (kép görgetése eltolással)	5 perc
3. Akvárium	Egyéni feladat a diákok számára. Nézzük meg egymás munkáit, beszéljük meg a lehetséges megoldásokat.	10 perc
4. Zenélő akvárium	Tanári bemutató, új funkciók megismerése (dallam lejátszása)	15 perc
5. Csillaghullás (egyéni munka)	Egyéni feladat a diákok számára. A végén a diákok nézzék meg egymás munkáit.	10 perc
6. Emeletes ház szimulációja	Tanári bemutató, új funkciók megismerése, ötletelés (a led kijelző koordinátáinak használata, pontok kigyújtása/kikapcsolása, állapotának ellentétesre állítása koordináta alapján)	15 perc
7. Emeletes ház továbbfejlesztés önállóan	Egyéni feladat a diákok számára. Nézzük meg egymás munkáit, beszéljük meg a lehetséges megoldásokat.	5 perc
8. Emeletes ház szimuláció továbbfejlesztése	Tanári bemutató, új funkciók megismerése (véletlenszerűség, eltolás, led kijelző ki/bekapcsolása)	15 perc
9. Emeletes ház továbbfejlesztés önállóan (5x5)	Egyéni feladat a diákok számára. Nézzük meg egymás munkáit, beszéljük meg a lehetséges megoldásokat.	10 perc
10. Érdekesség		

1. Ismétlés

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



Kezdő állapot

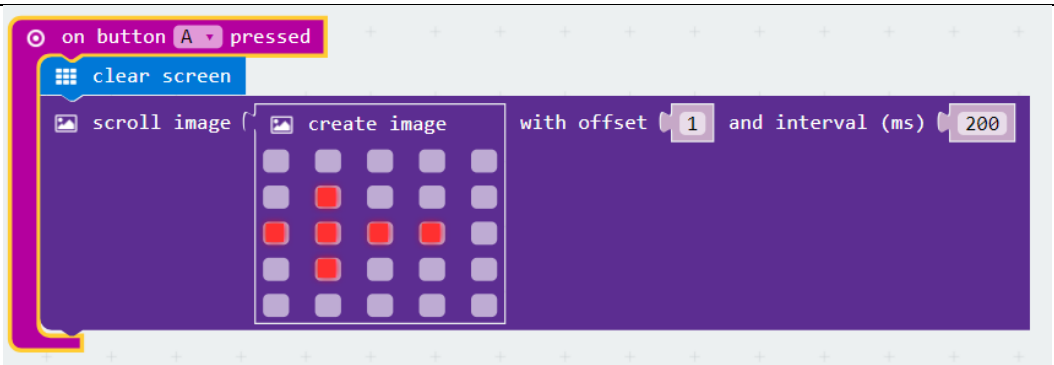


Végső állapot

2. Animáció eltolással

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_DR4AXEcDkHA2

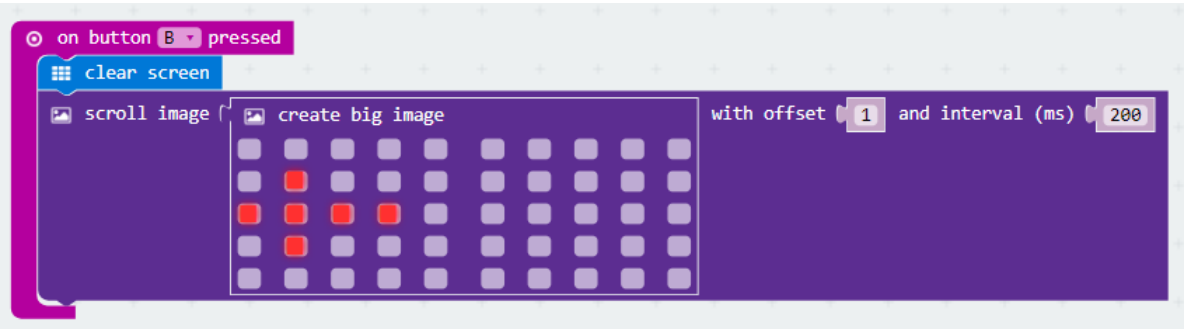
A diákok által elkészített animáció sok manuális munkát igényelt, mivel fázisról fázisra kellett megrajzolni a nyíl mozgását. Mutassuk meg, hogy ezt a feladatot hogyan oldhatjuk meg hatékonyabban a `led` kategória `scroll image with offset and interval (ms)`, valamint `create image` blokkjaival.



A fenti program a beállított képet görgeti jobbról balra, a megadott eltolással (*offset*) és késleltetéssel. Ez a megoldás olyan esetben lehet jó, ha egy adott ábrának kell jobbról balra, vagy balról jobbra mozognia. Utóbbi esetben az eltoláshoz negatív számot kell megadnunk.

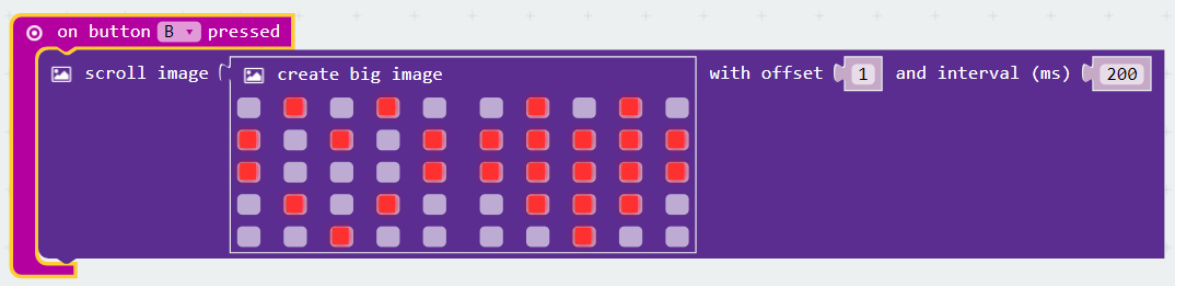
Ugyanezt nagyobb képpel (10 pont széles) is megcsinálhatjuk. Próbáljuk ki a következőt!

Illesszük be a nagyobb képet, és a bal oldalára rajzoljuk meg a nyilat, a jobb oldalát hagyjuk üresen.



Mit tapasztalunk? A nyíl kirepül bal oldalon. Beszéljük meg, hogy ez miért történt, mi ennek a magyarázata? (nyilván az, hogy a jobb oldali üres területet balra toljuk, így a nyíl fokozatosan eltűnik a kijelzőről.

Rajzoljuk meg az alábbi ábrát is. Az első 5x5-ös blokkba rajzoljunk egy üres szívet, a másodikba pedig egy teli szívet.



Próbáljuk ki, hogy mi történik az eltolás növelésével. Mi történik, ha elérjük az 5-ös számot az eltolásnál? Beszéljük meg közösen a tapasztaltakat! (Az ötös szám esetén ugyanazt a hatást érjük el, mintha a két képet egymás után játszanánk le.)

3. Akvárium

Feladat a diákok számára

Készíts egy olyan animációt a kép eltolásos 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 ússzon be egy még nagyobb hal, és tűnjön el a jobb oldalon.

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

4. Zenélő akvárium

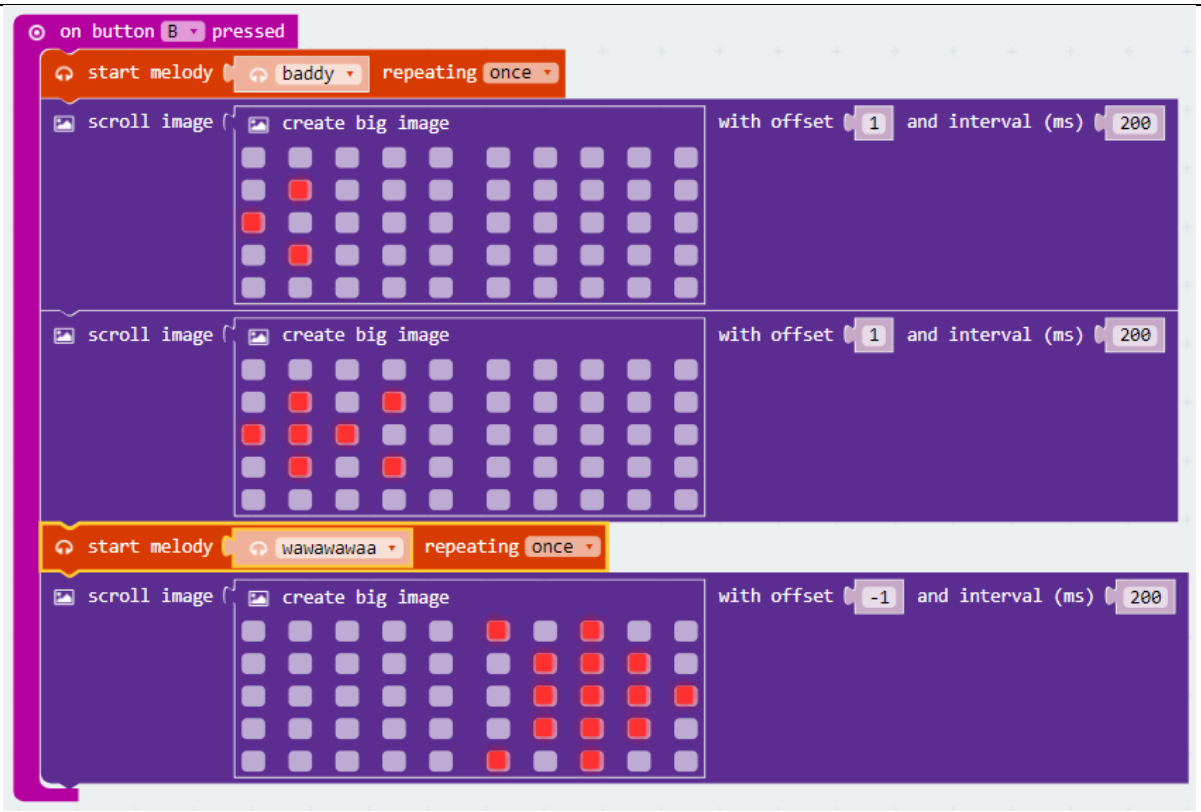
A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_cqCWCFgKug1X

A micro:bit nem csak animációkra képes, hanem zenét is képes lejátszani. Amennyiben a készletünkben van krokodil csipesz, vagy audió kábel és fülhallgató, akkor közvetlenül a micro:bit tudja szolgáltatni a zenét. Ha nincs, akkor a szimulátorban tudják a gyerekek kipróbálni a lehetőségeket.

Az előbb elkészült akvárium animációt (https://makecode.microbit.org/_Xe77oz4b42qe) egészítsük ki zenével. A **Music** kategóriából válasszunk ki **a start melody …. repeating … blokkot**, és válasszunk az animációhoz passzoló zenét, hanghatást.

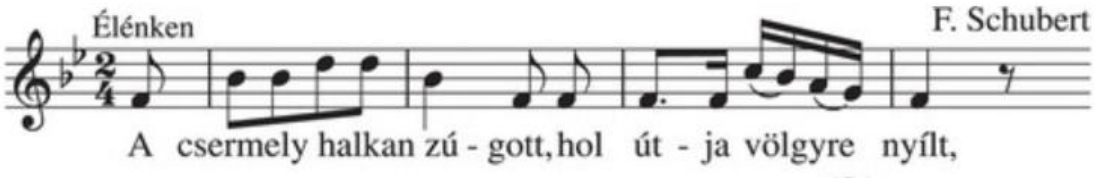
Például így:



Amint látható a kép görgetése előtt elkezdünk lejátszani egy dallamot. Majd a második görgetés után egy másik dallamot játszunk le.

Hagyjunk időt arra, hogy a diákok kipróbálhassák a zenei effekteket és elkészíthessék a saját zenés animációjukat.

Továbbfejlesztés: Valószínűleg a diákok nagy része hallotta már Schubert: A pisztráng című művét, amelynek kottájából származik az alábbi részlet.

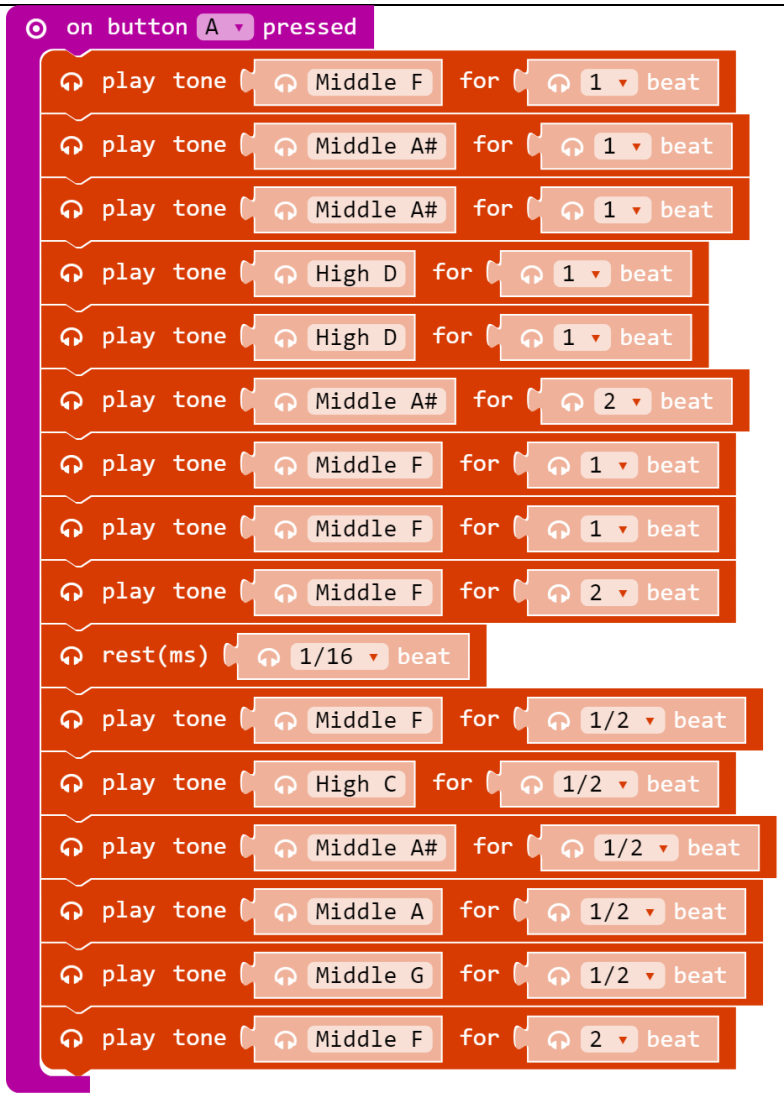


Forrás: http://www.mozaweb.hu/Lecke-ENK-Enek_Zene_7-A_pisztrang-99395

Próbáljuk meg ennek a műnek az első sorát lejátszani. Előtte viszont hallgassuk meg a mű elejét közösen: https://www.youtube.com/watch?v=kEiWXGA55PU

Az animáció kerüljön a `forever` blokkba, a zene pedig az „A” gomb lenyomásakor játszódjon le. A kottát nem tudjuk maradéktalanul átültetni, az alábbi kódrészlet nagyjából jól visszaadja az eredeti dallamot.

Programozzunk micro:biteket!



Hogy ne kelljen a diákoknak egyenként beállítani a hangokat, osszuk meg velük a projektet az alábbi webcímmel (https://makecode.microbit.org/_LeHHTpaDcXi6), amelyet így ők is kipróbálhatnak.

5. Csillaghullás

Feladat a diákok számára

Milyen hanghatással lehetne aláfesteni egy zuhanó tárgy mozgását? Készíts egy csillaghullás animációt, amelyben 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.

6. Emeletes ház szimuláció

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_M4ULeogwWW2k

A LED mátrix pontjait egy koordinátarendszer segítségével is kigyújthatjuk, illetve kiolthatjuk. Ehhez a **LED** kategória blokkjait kell használnunk. A következő projektben ennek módját mutatjuk be a diákoknak

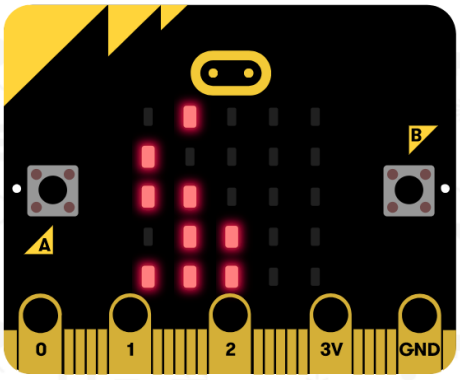
Tegyük fel a kérdés a diákoknak, hogy szerintük mihez hasonlít a micro:bit LED kijelzője? Sokféle választ kaphatunk (pl. dobókocka, Rubik-kocka, négyzetrácsos füzet, emeletes ház). Ha az emeletes ház nem jönne elő az ötleteléskor, vezessük rá a gyerekeket azzal a kérdéssel, hogy „*Ha a kis Ledek ablakok lennének, akkor mihez hasonlítana a kijelző?*”

Játszunk el a gondolattal, hogy a micro:bit kijelzője egy emeletes házat jelképez (földszint + 4 emelet). Mutassunk be olyan videót a gyerekeknek, amelyen az látszik, hogy egy emeletes ház fényei kigyúlnak, elsötétednek:

- https://www.youtube.com/watch?v=lt7sq7Ld-4g
- https://www.youtube.com/watch?v=M-34aBULH8g

Valami hasonlót fogunk most elkészíteni a micro:bit segítségével.

A feladat: Van egy 5 szintes toronyház, minden szinten 3 helyiséggel. A projektben azt fogjuk szimulálni, hogy a lakók este hazatérnek (véletlenszerű időpontban) és felkapcsolják a villanyt, valamint véletlenszerű időpontban lekapcsolják azt. A ház földszintjén állandóan fel legyenek kapcsolva a villanyok (itt éjjel nappal nyitva tartó üzletek üzemelnek).



Kérdés a diákokhoz: Egy emeletes ház lakásait hogyan szokták számmal azonosítani? Beszélgessünk erről picit. Ha valaki emeletes házban lakik a diákok közül, mi a lakásuk postacíme?

Válasz: Legtöbbször a címet úgy adják meg, hogy 1. emelet 2-es lakás, 4. emelet 3-as lakás, és így tovább. Vagyis külön hivatkoznak az emeletre, és azon belül kap egy sorszámot a lakás.

Az alábbi ábrák segítségével magyarázzuk el, hogy hogyan tudjuk számokkal (koordinátákkal) azonosítani az egyes lakásokat egy emeletes házban.

1	2	3	4. emelet
1	2	3	3. emelet
1	2	3	2. emelet
1	2	3	1. emelet
1	2	3	földszint

Ha megnézzük a bal oldali emeletes házat, mindegyik szintnek van egy azonosítója, illetve minden lakásnak is.

Ha meg akarunk jelölni egy lakást, mondhatjuk azt, hogy 3. emelet 1. lakás, vagy földszint 3-as lakás.

1. lakás	2. lakás	3. lakás	
1,4	2,4	3,4	4. emelet
1,3	2,3	3,3	3. emelet
1,2	2,2	3,2	2. emelet
1,1	2,1	3,1	1. emelet
1,0	2,0	3,0	földszint (0. emelet)

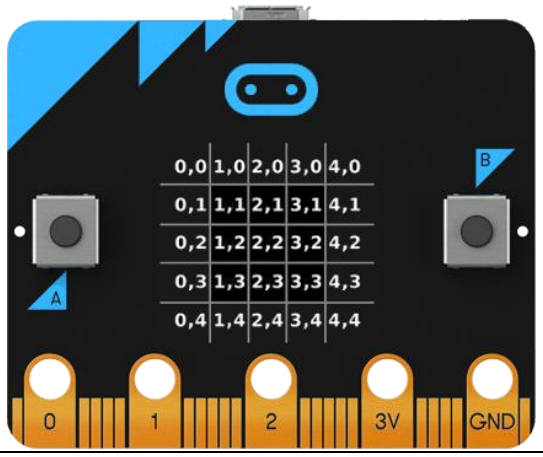
Kérdés? Hogy tudnánk számsorokkal jelölni az egyes lakásokat úgy, hogy mindegyikről tudjuk, hogy melyik szinten van, és hányadik lakás?

Például így, ahogy a bal oldali ábrán látjuk. Az első szám a lakás sorszámát jelentené, a második szám pedig az emelet számát, vagy nullát, ha földszintről van szó.

Ezek után térjünk át arra, hogy a micro:biten hogyan tudjuk az egyes pontokat azonosítani. Az elv hasonló lesz, de itt a sorokat és az oszlopokat is nullától kell sorszámozni, illetve fontos különbség, hogy nem a legalsó sor lesz a nullás sorszámú, hanem a legfelső, és a sorok száma lefelé növekszik. Az első oszlop lesz a nulladik sorszámú, és a számok jobbra növekednek.

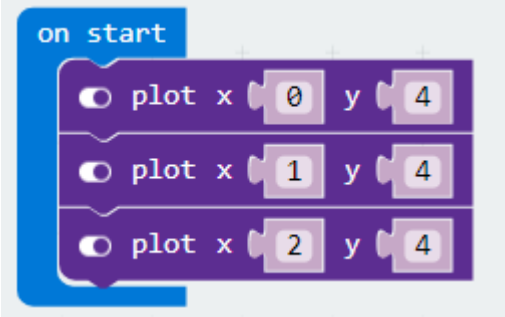
A LED kijelzőn az oszlopokat az X betű fogja azonosítani, a sorokat pedig az Y betű. (Ezeket koordinátáknak nevezzük). Az alábbi ábrákon feltüntettük a pontok koordinátáit, néhány nevezetes pont koordinátájának kiemelésével.

	X	Y
Bal felső sarok	0	0
Jobb felső sarok	4	0
Középső pont	2	2
Bal alsó pont	0	4
Jobb alsó pont	4	4

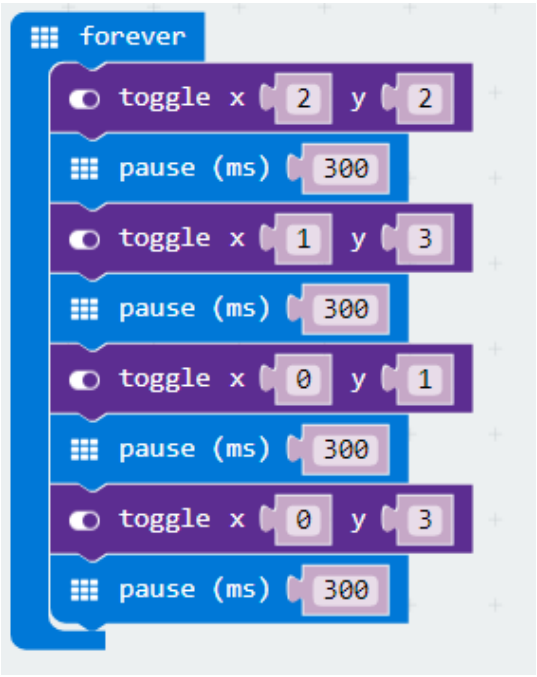


Most, hogy ismerjük a koordináták jelentését, nézzük meg, hogy hogyan tudjuk kigyújtani a Led kijelző adott koordinátájú pontját. Ehhez a `plot x y` blokkot használhatjuk. Ha pedig egy pontot ki akarunk oltani, akkor az `unplot x y` blokkot kell használnunk.

Kezdjük el a projektet közösen megvalósítani!



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



Ezek után néhány pont állapotát változtassuk meg az ellenkezőjére. Erre szolgál a `Led` kategória `toggle x y` blokkja. A váltások között várakozzunk picit a korábban használt `pause` blokk segítségével.

A megoldás bal oldalon látható.

7. Emeletes ház (továbbfejlesztés)

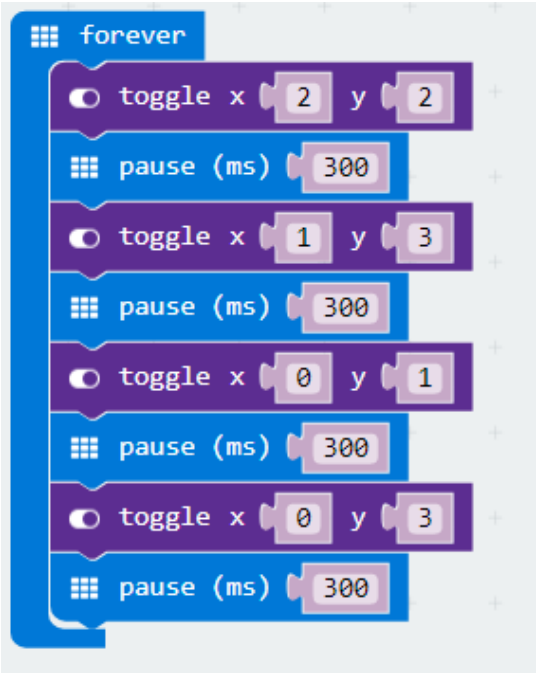
Feladat a diákok számára

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

A lámpák kioltása ugyanilyen sorrendben történjen.

8. Emeletes ház szimuláció továbbfejlesztése (véletlenszerűség, eltolás, Led kijelző ki/bekapcsolása)

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_07iL5qMK0e56
Milyen jó lenne, ha az ablakokat véletlenszerűen tudnánk kiválasztani. Sokkal életszerűbb lenne az animációnk. A következőkben ennek módját mutassuk be a diákoknak.

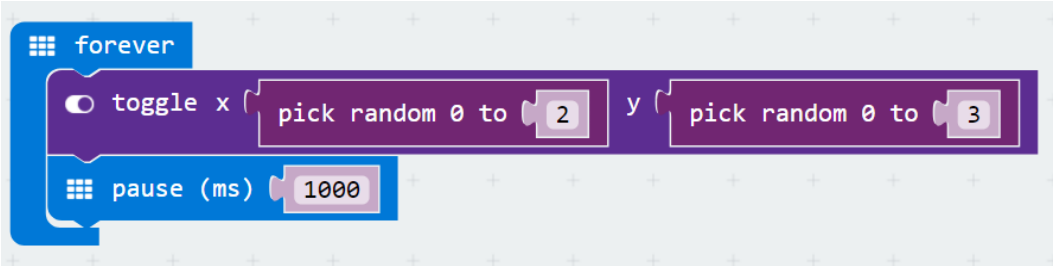


Korábban mi határoztuk meg, hogy mely pontokat gyújtjuk ki. Most ugyanezt a micro:bitre bízzuk.

Ehhez a Ehhez a **math** kategória **pick random 0 to x** blokkját kell használnunk. Ez a blokk azt jelenti, hogy 0 és a megadott szám között egy véletlenszerűen választottat fogunk kapni eredményül.

Az X (oszlop) koordináta esetén 0 és 2, az Y (sor) koordináta esetén 0 és 3 között kell véletlenszámot előállítanunk.

Vagyis a megoldás:



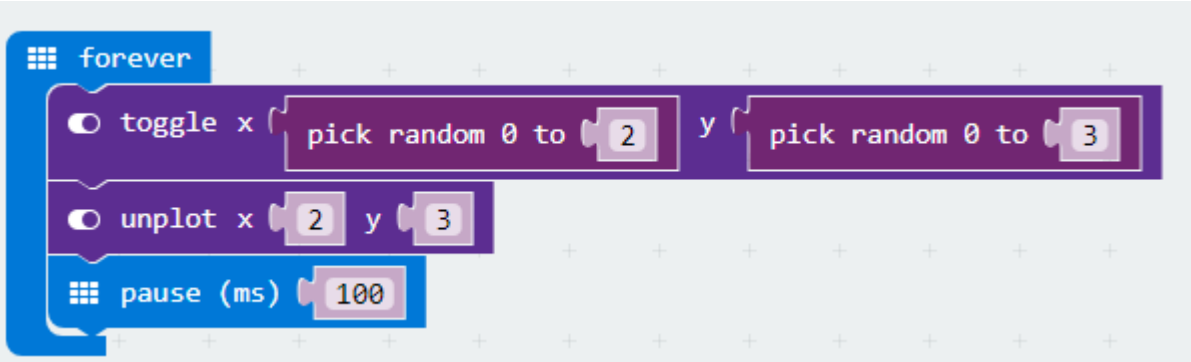
Továbbfejlesztés:

A Kovács család elutazott, ezért az Ő lakásuknak sötétnek kell maradnia. Ők az 1. emelet 3. lakásában laknak. Beszéljük meg együtt, hogy mi ennek a lakásnak a koordinátája. Megoldás: (X=2;Y=3).

Hogyan oldhatjuk meg az eddigi tudásunkkal, hogy sötét legyen ez a lakás? Ötleteljünk közösen!

A mélyebb előismeretekkel rendelkező diákok valószínűleg javasolni fogják, hogy elágazást készítsünk, és csak akkor engedjük kigyújtani a pontot, ha az különbözik a 2;3 koordinátától. Ez a megoldás is jó lehetne, de most mutassunk egy egyszerűbb megoldást. A **toggle** (átkapcsolás) művelet után mindenképpen oltsuk ki ennek a lakásnak a lámpáját az **unplot** blokk segítségével.

Ezen változat linkje: https://makecode.microbit.org/_Hve5kvDzJ7ra



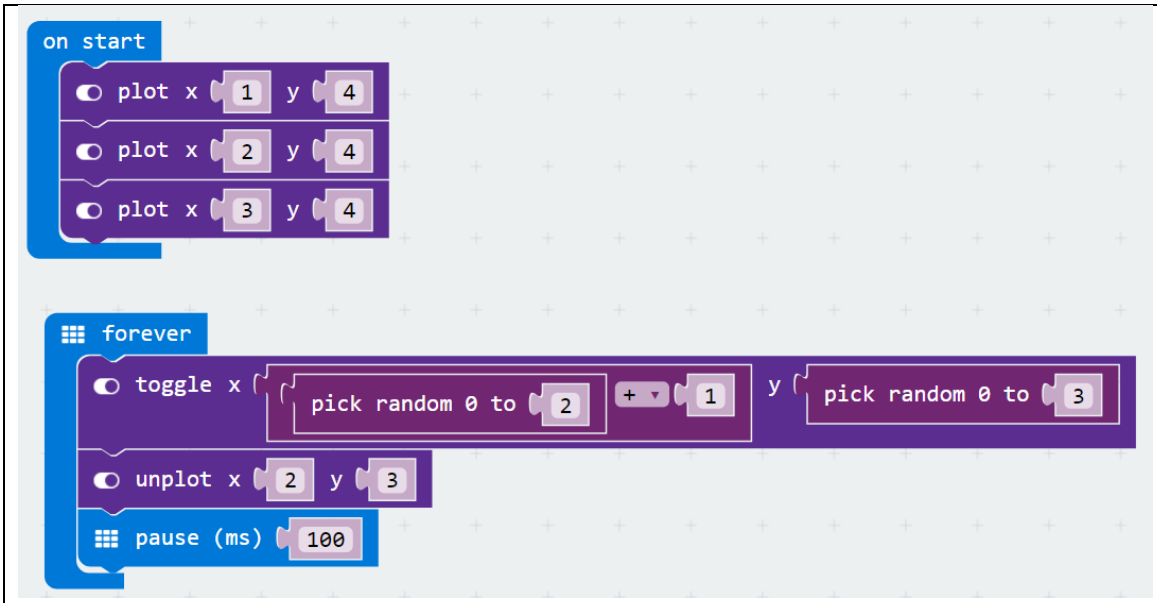
Továbbfejlesztés: Toljuk el a toronyházat eggyel jobbra, hogy középre kerüljön. Kérdezzük meg a diákokat, hogy ezt hogyan lehetne megoldani?

Valószínűleg többen azt javasolják majd, hogy növeljük meg a véletlenszám generátor paraméterét, de ez nem megoldás, hiszen így nagyobb épületet készítenék, nem pedig eltolnánk azt. A helyes válasz, hogy a kapott véletlenszámot meg kell növelnünk.

Készítsük el azt a változatot, amelyben az X koordináta értékét eggyel megnöveljük. Ehhez a `math` kategória `összeadás` blokkját fogjuk felhasználni.

Az így módosított projekt elérhetősége:
https://makecode.microbit.org/_3dybH96xg4Dy

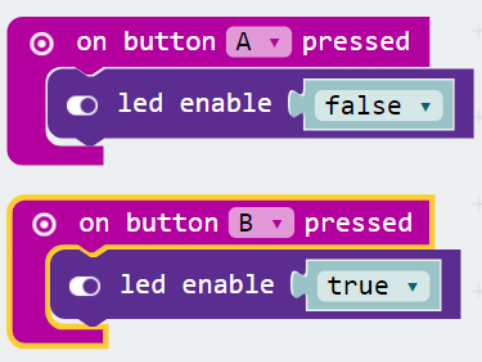
A megoldás:



Továbbfejlesztés

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

Erre a célra a **Led** kategória **led enable true/false** blokkját kell használnunk.



A teljes megoldás elérhetősége: https://makecode.microbit.org/_Uz19UbEbqb8Y

Érdekesség bemutatása: Egy emeletes ház ablakaival akár ábrákat, animációkat is meg lehet jeleníteni. Ehhez szükség van olyan vezérlő egységre, ami megfelelő időpontban bekapcsolja és kikapcsolja a lámpákat. Vagyis amit kicsiben a micro:bittel megvalósítottunk, azt akár nagyban is meg lehet valósítani. Az alábbi videót érdekességképpen mutassuk be a diákoknak: https://youtu.be/b4oIf4v3Sbk

A videón a BME egyik kollégiuma látható, amelyet a diáknapokon óriáskijelzővé alakítanak. A szobákban színes lámpákat helyeznek el, így még élvezetesebb animációkat tudnak megjeleníteni az épület ablakainak segítségével.

9. Emeletes ház (5x5)

Feladat a diákok számára

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

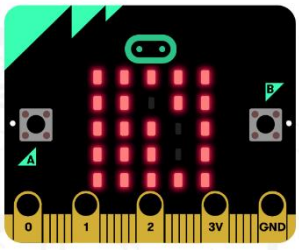
3. alkalom (Rajzok/animációk megjelenítése a LED mátrixon)

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Varázsdoboz	Egyéni feladat a diákok számára. (ismétlés)	15 perc
2. Rajzok/Animációk a világosság-érték módosításával	Tanári bemutató, új funkciók megismerése (led fényességének beállítása)	5 perc
3. Animáció ciklus segítségével	Tanári bemutató, új funkciók megismerése, ötletelés (számlálós ciklus használata, matematikai alapműveletek)	10 perc
4. Különböző világosságú csíkok	Egyéni feladat a diákok számára. (A feladat nyomtatva kiadható, vagy kivetíthető)	10 perc
5. Animáció több eszközön (ötletelés és megvalósítás)	3-4 fős csoportok alakítása. A gyerekek elkészítik az animáció forgatókönyvét. A tanár válaszol a felmerülő kérdésekre, segíti a projektek megvalósulását.	20+20 perc
6. Bemutató	A csoportok által készített animációk megtekintése. Érdemes a munkákat videóra felvenni, és az alkotásokat megosztani, például az iskola honlapján, vagy közösségi oldalán.	10 perc

1. Varázsdoboz

Feladat a diákok számára

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 oltódjon 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 elfeketíteni a LED kijelzőt, a „B” gombbal pedig visszakapcsolni azt.

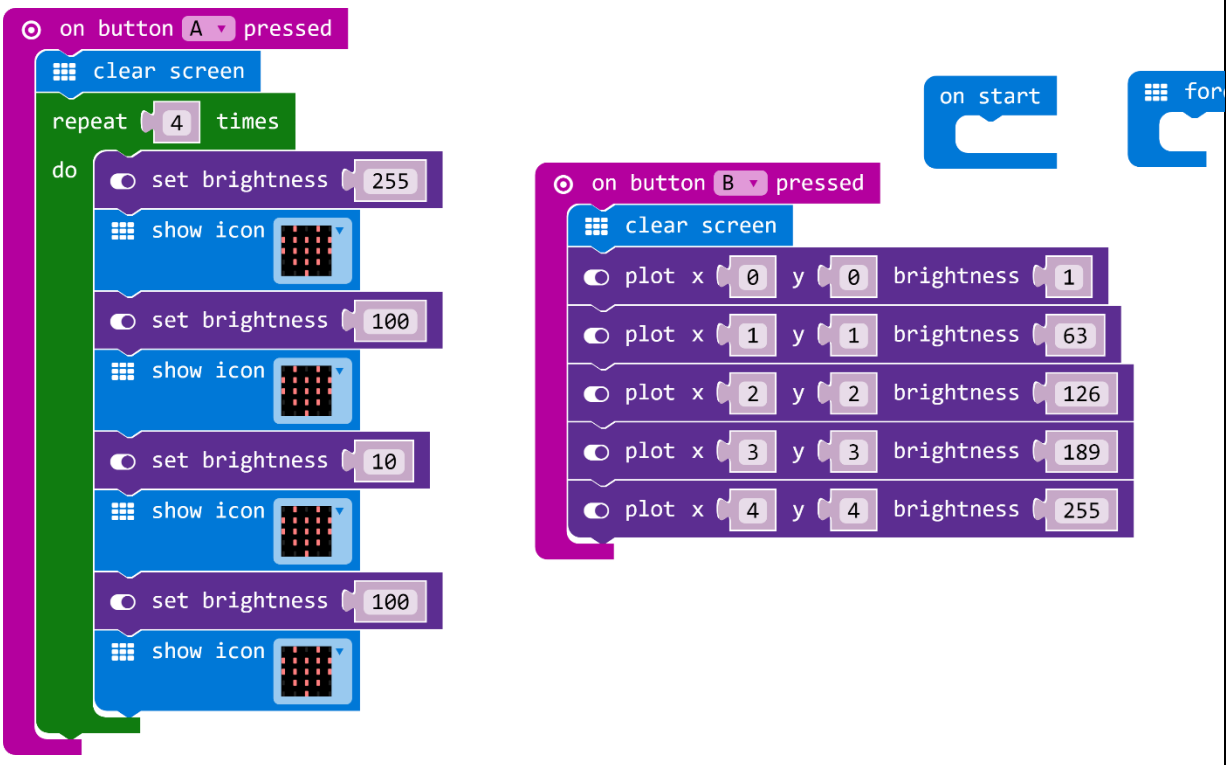
2. Rajzok/Animációk a világosságérték módosításával

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_MAzAdbMyrDKA

A micro:bit kijelzőjén megjelenő ábrának világosságértéke is beállítható. Ehhez a **led** kategória **set brightness** blokkját kell használni. Az érték 0 és 255 között vehet fel értéket, a 0 a teljes sötétséget, a 255 a teljes világosságot jelenti.

Mutassuk be a következő egyszerű projektet a fenti link segítségével! A linket osszuk meg a diákokkal is, hogy ki tudják próbálni a projekt eredményét.



Az „A” gomb lenyomásával egy szívdobogást tudunk bemutatni, ahol az egyes fázisok csak a világosságértékben különböznek.

A „B” gomb megnyomásakor egy átlós szakasz jelenik meg folyamatosan erősödő világosságértékkel. Ezt a **Led** kategória **plot x y brightness** blokkjával tudjuk megvalósítani. Ez utóbbi pl. a szimulátor ablakban nagyon szépen látszik, a micro:bitre

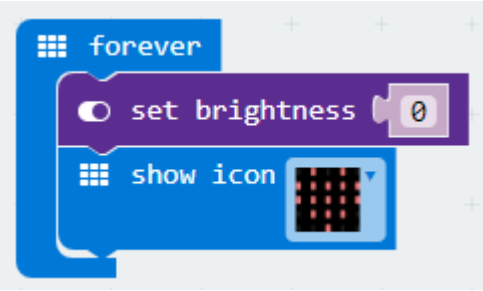
töltve nem biztos, hogy érzékeljük a különbséget, vagy hunyorítanunk kell ahhoz. Ezért mindenképpen javasolt, hogy nagy világosságérték különbségeket állítsunk be, ha tényleg látni akarjuk a különbséget!

3. Animáció ciklus segítségével

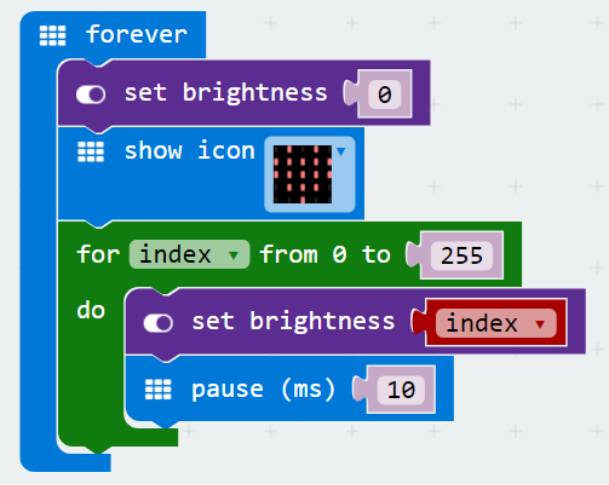
A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_A5xd30Jj5CX3

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 megtanuljuk, hogyan egyszerűsíthetjük a feladatot ciklus használatával.

Készítsük el közösen az alábbi egyszerű projektet, amelyben beállítjuk a fényerősséget nullára, majd kirajzolunk egy ikont.



Most fokozatosan állítsuk be a fényerősség szintjét 0-tól egészen 255-ig. Ehhez számlálós ciklust kell használnunk a következőképpen:



A **Loops** kategóriából válasszuk ki a **for** számlálós ciklust, és állítsuk be, hogy 255-ig menjen a ciklus. Magyarázzuk el a diákoknak, hogy mit jelent a ciklus (mi a szerepe az index-nek, miért hívjuk ezt változónak) Magyarázzuk el, hogy az index változót hogyan tudjuk felhasználni a világosságérték beállításánál (a variables kategóriában megtaláljuk, csak be kell húzni az érték helyére). Hagyjunk időt arra, hogy mindenki kipróbálhassa a projektet.

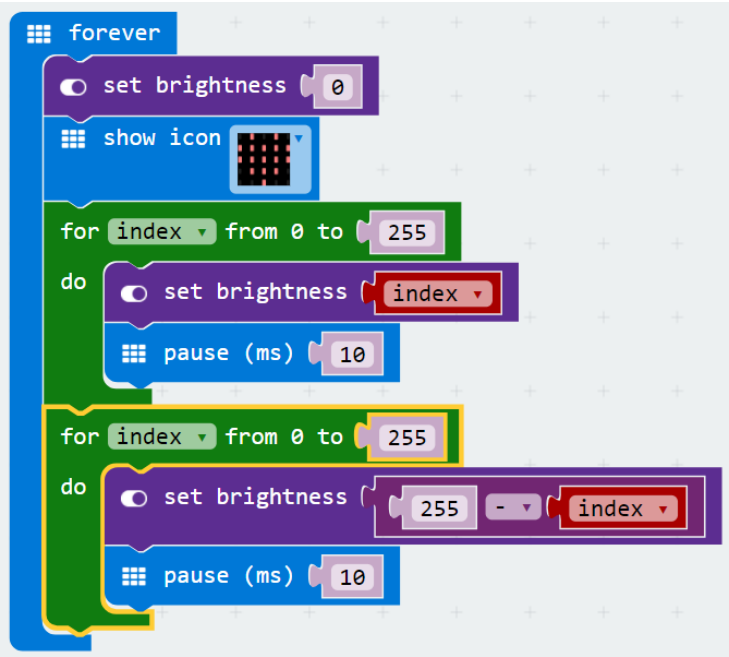
Továbbfejlesztés: Visszaszámlálás

Hogyan tudnánk megoldani, hogy visszafele (255-től 0-ig) állítsuk be a világosságértéket. Vessük fel a problémát a diákoknak, ötleteljünk közösen. Segítségképpen írjuk fel a két számsorozatot a táblára.

0	1	2	3	4	5	...	255
255	254	253	252	251	250	...	0

A felsőt már elő tudtuk állítani. A kérdés, hogy a felső számsorozatból hogyan állítható elő az alsó?

A megoldás, ki kell vonni 255-ből a fenti sorozatot, így pont az alsót kapjuk, vagyis egy visszaszámlálást tudunk megvalósítani. Ehhez a **Math** kategória **kivonás** blokkját is fel kell használnunk az alábbi módon.

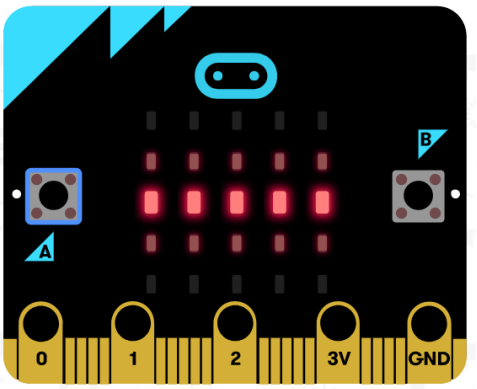


4. Különböző világosságú csíkok

Feladat a diákok számára

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` kategória `plot x y brightness` blokkját kell használni.

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



A pontok balról jobbra, egymás után, kis késleltetéssel gyúljanak ki az „A” gomb megnyomásakor.

5. Animáció több eszközön (ötletelés)

A tanár által /bemutatandó mintaprojekt

https://www.youtube.com/watch?v=Q1XEH-rjYH4 (2 perc 27 másodperc)

Játsszuk le a fenti videót a diákok számára. Ezen az látható, hogy több okostelefonon egyszerre indul el az animáció, az eszközök egymás mellé rakásával remek animációk készíthetőek, ehhez viszont össze kell hangolni az animációkat.

Micro:bitekkel ilyen kidolgozottságú animációkat természetesen nem tudunk készíteni, de azért nem kell lebecsülni sem az eszköz képességeit, sem a diákok kreativitását.

Alakítsunk 3-4 fős csoportokat. A csoportok feladata az lesz, hogy készítsenek olyan animációt, amelyhez 3-4 micro:bit képességeit használják fel. Kezdjék el az ötletelést, a forgatókönyv készítését, a megvalósítás a következő alkalomra marad.

Ügyelni kell arra, hogy a megvalósítandó projekt ne legyen túl bonyolult, meg lehessen valósítani a következő alkalommal, de túl egyszerű se legyen. Ha már szó esett a micro:bit zenei képességeiről is, érdemes lenne az animációkat zenével kísérni.

Feladatok, ha van rá idő

Amennyiben a projektek megvalósítása után marad még idő, kiadhatjuk az alábbi gyakorló feladatokat is:

Feladat a diákok számára

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

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

3. 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 előadó neve.

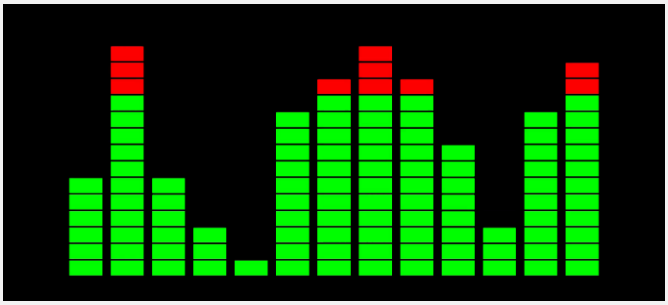
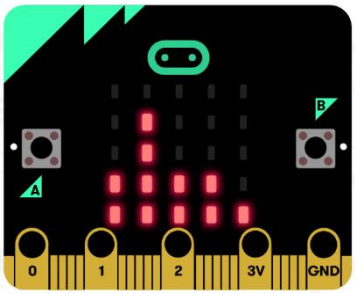
4. alkalom Egymásba ágyazott ciklusok, elágazások használata, változó értékének növelése, csökkentése

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Ismétlő feladat	Önállóan megoldandó, ismétlő feladat a diákok számára (pontok véletlenszerű kigyújtása és kioltása).	5 perc
2. Jelszint kijelző szimuláció	Tanári bemutató, új funkciók megismerése, ötletelés (új változó létrehozása, értékének beállítása, egymásba ágyazott számlálós ciklusok alkalmazása)	15 perc
3. Jelszint kijelző szimuláció, továbbfejlesztés, egymásba ágyazott ciklusok	Tanári bemutató, új funkciók megismerése (változó létrehozása, értékének beállítása, egymásba ágyazott ciklusok használata)	5 perc
4. Növekvő fényerő	Önállóan megoldandó feladat a diákok számára a korábban bemutatott funkciók felhasználásával.	15 perc
5. Elágazás használata, különböző feltételek kipróbálása	Tanári bemutató, új funkciók megismerése (elágazás használata, logikai feltételek megadása)	15 perc
6. Elágazás használata, különböző feltételek kipróbálása – önálló feladat	A diákok a feladatul adott ábrákat próbálják előállítani a feltételek megváltoztatásával. Hagyjunk időt a próbálkozásra, majd beszéljük át a megoldásokat. Gyűjtsük össze a diákok által készített érdekesebb ábrákat, és azokból is válogathatunk közös gondolkodáshoz.	15 perc
7. Programajánló alkalmazás (elágazás különben ággal)	Tanári bemutató, új funkciók megismerése (elágazás különben ággal).	5 perc
8. Kő, papír, olló játék	Önállóan megoldandó feladat a diákok számára a korábban bemutatott funkciók felhasználásával.	5 perc
9. Kő, papír, olló játék – pontok nyilvántartása	A diákok által készített projekt közös továbbfejlesztése a pontszámok nyilvántartásához. A játék kipróbálása.	10 perc

1. Ismétlő feladat

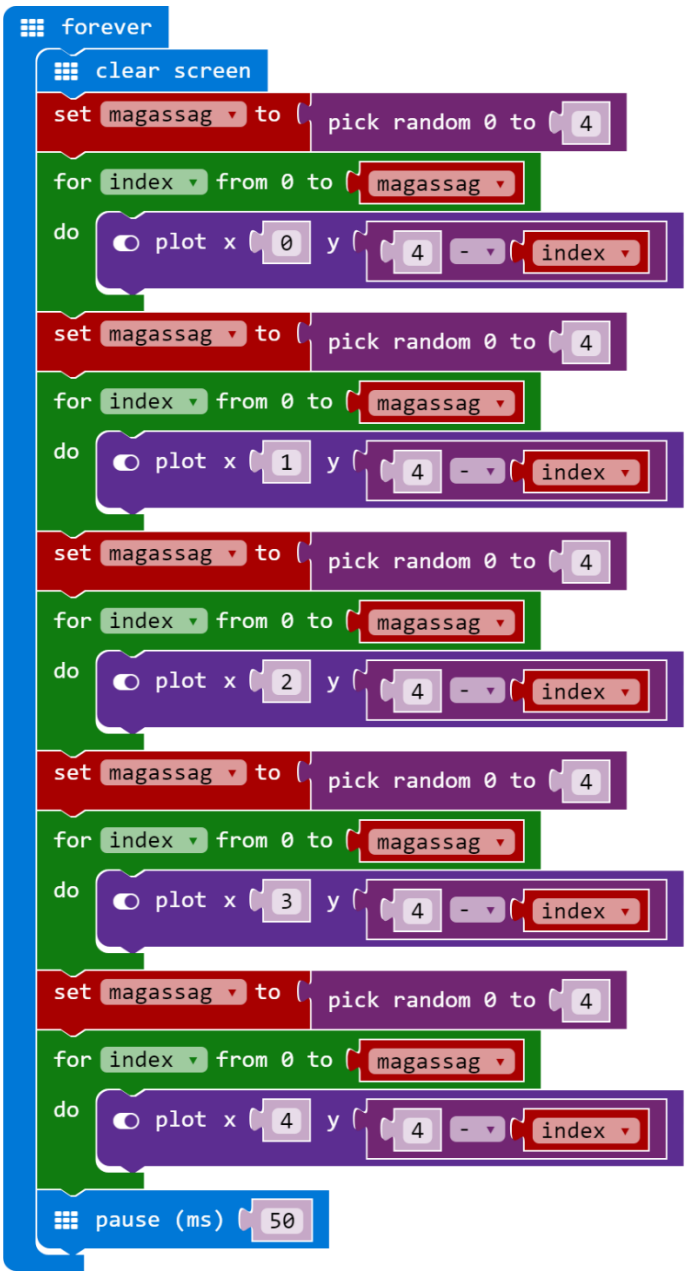
Feladat a diákok számára
Készíts programot, amely elhelyez 100 pontot véletlenszerűen választott koordinátán, majd letöröl a kijelzőről 200 véletlenszerűen választott pontot és ezt a két műveletet folyamatosan ismételgeti

2. Jelszint kijelző szimuláció

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_iPLf8uMFo9pc
 HIFI eszközökön, zenelejátszó alkalmazásokon gyakran látunk olyan kijelzőt, amely a hang erősséget jelzi ki különböző tartományokban. A következőkben olyan alkalmazást fejlesztünk, amely ezt szimulálja.
Kezdésképpen mutassuk be a https://youtu.be/vWpC5PtNE6E videót, amelyen jól látható, hogy miről van szó.  Mi a kijelző oszlopaiban (véletlenszerűen meghatározott) eltérő magasságú pontsorok rajzolásával fogjuk ezt a működést szimulálni. Ahhoz, hogy ez működjön, használnunk kell a korábban tanultakat (rajzolás a Led kijelzőre, ciklusok, véletlenszámok), de szükség van új ismeretekre is, például arra, hogy hogyan hozzunk létre változót és adjunk neki értéket.

Mutassuk be, hogy hogyan hozhatunk létre új változót (**Variables** kategória, **Make a variable**). Hozzuk létre a magasság nevű változót, ez fogja tartalmazni, hogy milyen magasságú pontsort kell kirajzolnunk az adott oszlopba. 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 kapott véletlenszám 4, akkor nem a legalsó sorban (4-es y koordinátára) fogunk rajzolni, hanem a legfelsőn, aminek az y koordinátája 0.

Készítsük el közösen az alábbi projektet. A blokkokat duplikáljuk, hogy gyorsan össze tudjuk állítani a projektet.



A feladatot kiegészíthetjük zene lejátszással is, mintha valóban zenére működne az alkalmazásunk. Érdemes fehívni a diákok figyelmét arra, hogy külső szenzorok, érzékelők csatlakoztatásával valóban működő, a hangra reagáló kijelzőt is lehetne készíteni.

3. Jelszint kijelző szimuláció, továbbfejlesztés, egymásba ágyazott ciklusok

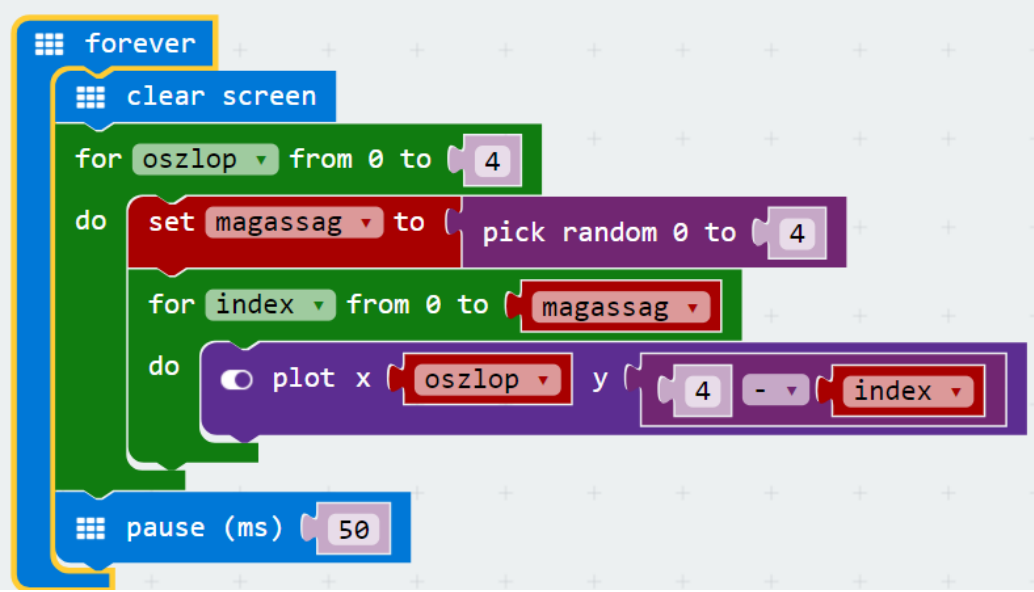
A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_cJrAXToLsKYv

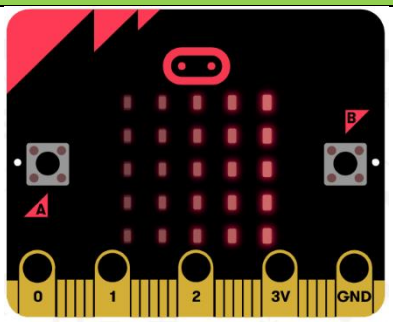
Az alkalmazásunk ugyan szépen működik, de a kódja nem eléggé hatékony. Hogyan tudnánk hatékonyabbá tenni?

Vegyük észre, hogy a kódunkban az egyes rajzolási blokkok alig különböznek egymástól, a különbség csak annyi, hogy a nulladik sorszámú, 1-es, 2-es, 3-as, vagy 4-es sorszámú oszlopban kell-e rajzolunk. Vagyis ezt a rajzolási blokkot akár ciklusba is szervezhetnénk.

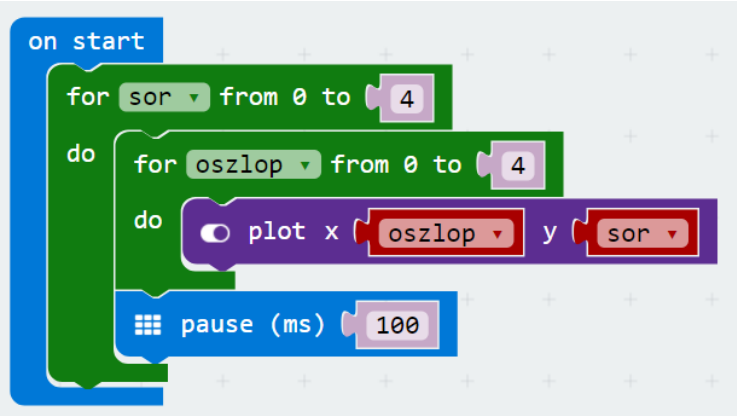
Előtte hozzunk létre egy oszlop nevű változót, ami azt tartalmazza, hogy éppen milyen koordinátájú oszlopban kell rajzolnunk.



4. Növekvő fényerő

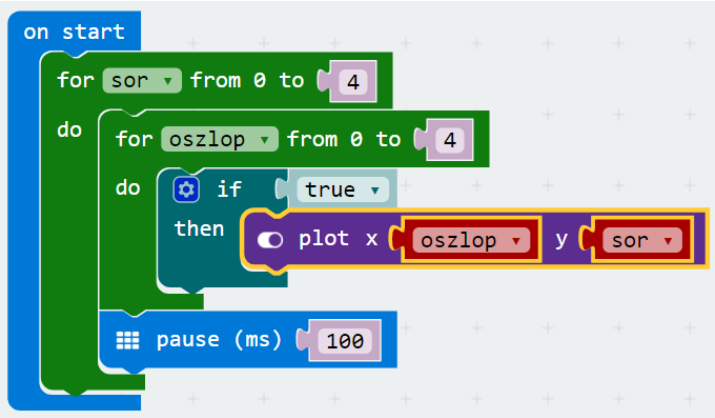
Feladat a diákok számára
Készíts olyan alkalmazást egymásba ágyazott ciklus használatával, amely a Led kijelző összes pontját kigyújtja soronként, balról jobbra haladva. 
Módosítsd az alkalmazást úgy, hogy az 1. oszlopban lévő pontok fényereje (*brightness*) legyen 1, a másodikban lévők 11, a harmadikban 21, és így tovább. Hogyan lehet a ciklusváltozó értékétől függővé tenni a világosságértéket?
Ha ezzel kész vagy fejleszd tovább úgy az alkalmazást, hogy a fenti rajzolás „B” gomb hatására történjen. Az „A” gomb megnyomásakor pont ellentétes legyen az ábra, vagyis az 1. oszlop legyen a legfényesebb, és fokozatosan csökkenjen a fényerő.

5. Elágazás használata, különböző feltételek kipróbálása

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_eF2KTxUwDcL8
Osszuk meg a diákokkal az alábbi projektet: https://makecode.microbit.org/_eF2KTxUwDcL8  Ez a program dupla ciklussal kigyújtja az összes pontot a Led mátrixon.

Módosítsuk a programot egy feltétel hozzáadásával az alábbi módon. A logic kategóriából válasszuk ki az **if true then** blokkot és abban helyezzük el a pont kigyújtásáért felelős **plot** blokkot. Magyarázzuk el, mit jelent az elágazás!

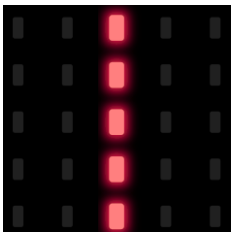
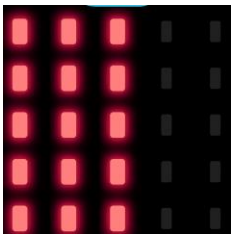
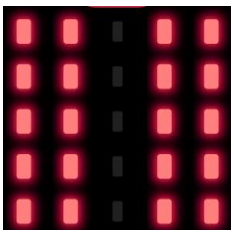
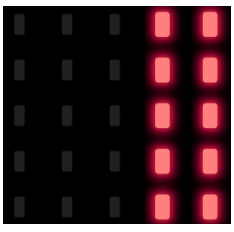
Próbáljuk ki az eredményt. Nem tapasztalunk változást, mivel a *true* azt jelenti, hogy igaz, az elágazás akkor ága pedig igaz esetén végrehajtódik, vagyis most mindig végrehajtódik a pont kirajzolás.

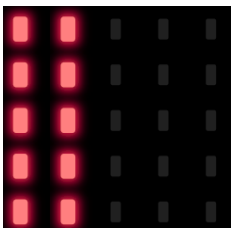
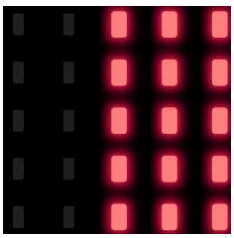


Változtassuk meg a feltételt az alábbi módon, szintén a **Logic** kategória 

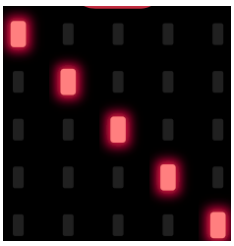
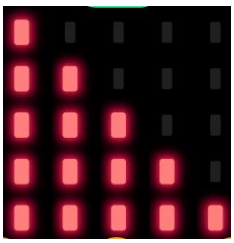
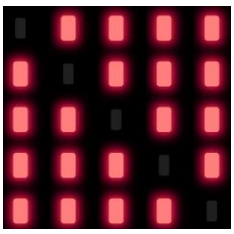
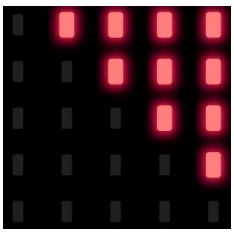
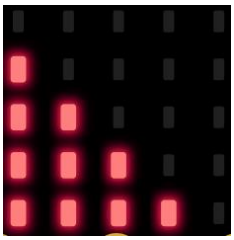
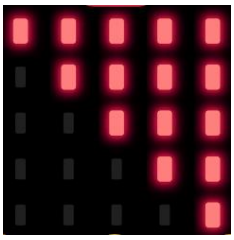
blokkjának behúzásával. Az első szám helyére húzzuk be az oszlop változót az egyenlőségjel után pedig írjuk a kettes számot! Mit tapasztalunk?

Ebben az esetben csak a második oszlop pontjai lesznek kigyújtva. Próbáljuk ki az összes relációjelet és minden egyes módosítás után beszéljük meg a diákokkal a kapott eredményt. (elég csak a szimulátort használnunk a bemutató során)

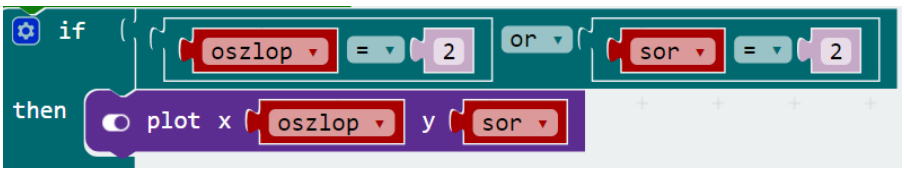
Feltétel	**Eredmény**	**Feltétel**	**Eredmény**
oszlop=2		oszlop≤2	
oszlop≠2		oszlop>2	

oszlop<2		oszlop≥2	

Próbáljuk ki azt is, hogy az oszlop és a sor változó értékét hasonlítjuk össze. Így átlós ábrákat fogunk kapni. Beszéljük meg a látottakat.

Feltétel	**Eredmény**	**Feltétel**	**Eredmény**
oszlop=sor		oszlop≤sor	
oszlop≠sor		oszlop>sor	
oszlop<sor		oszlop≥sor	

Most próbáljuk ki a **vagy** logikai kifejezést is az alábbi módon:

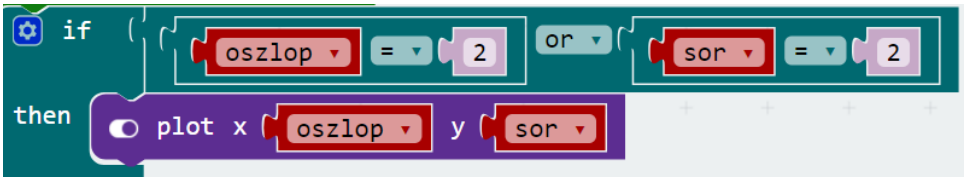


Beszéljük meg, hogy miért jelenik meg egy kereszt a képernyőn!

A vagy helyett használjunk **és** (and) kapcsolatot. Most mit tapasztalunk? (megoldás: csak a középső pont lesz kigyújtva, mivel arra igaz, hogy a sor és oszlop koordinátája is 2.)

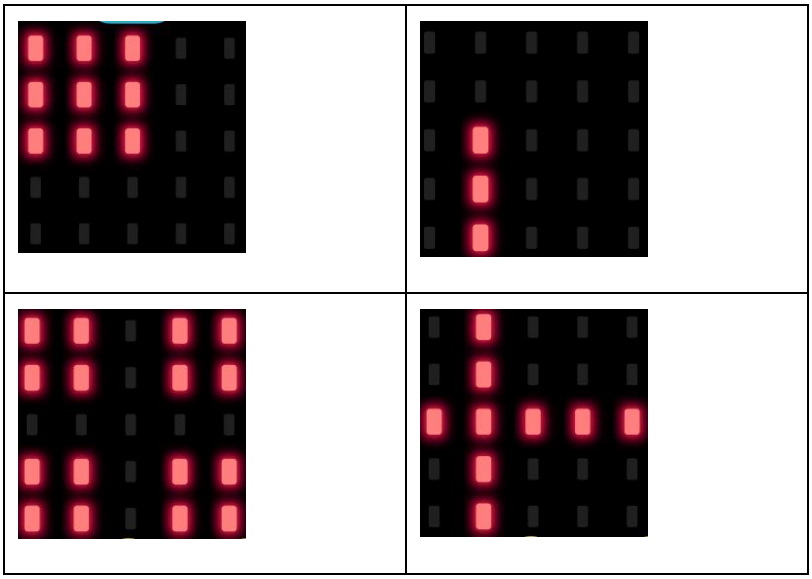
6. Elágazás és feltételek – önálló feladat

Feladat a diákok számára



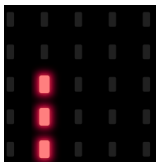
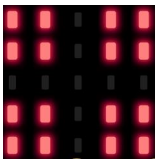
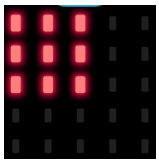
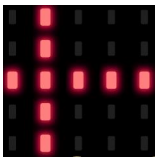
Milyen feltételeket kell megadni ahhoz a fenti blokkok módosításával, hogy az alábbi ábrákat kapjuk eredményül? (csak a legördülő értékeket kell megváltoztatni a fenti blokkban, más módosítás nem szükséges)

Ügyelj arra, hogy az oszlopok és sorok 0-tól sorszámozódnak. Próbáld ki, hogy tényleg megfelelő ábrát kapsz-e!



Ha kész vagy a fentiekkel, kísérletezz nyugodtan a feltételek megváltoztatásával, és ha kapsz egy érdekes ábrát, mutasd meg a tanárodnak és jegyezd fel, hogy hogyan sikerült előállítanod.

Megoldás

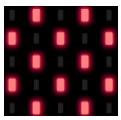
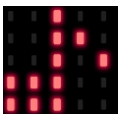
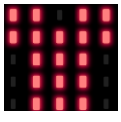
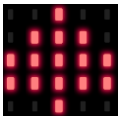
Ábra	Feltétel	Ábra	Feltétel
	(oszlop=1) és (sor>1)		(oszlop≠2) és (sor≠2)
	(oszlop≤2) és (sor≤2)		(oszlop=1) vagy (sor=2)

7. Programajánló alkalmazás (elágazás készítés különben ággal)

A tanár által elkészítendő/bemutatandó mintaprojekt

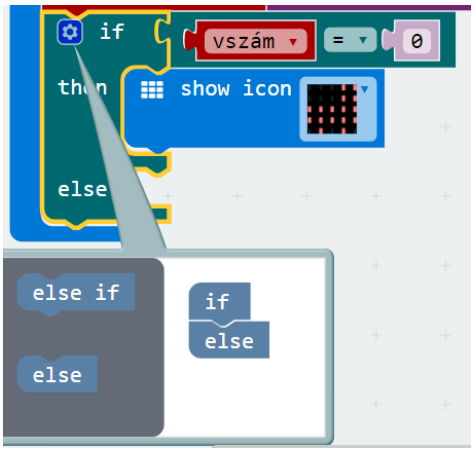
https://makecode.microbit.org/_bJt8pkWqRTgo

Készítsünk egy alkalmazást, amely programot ajánl nekünk, ha unatkoznánk. Az egyes szabadidős programokból véletlenszerűen válasszon az alkalmazásunk, és ikonokkal jelezze, hogy mit kellene csinálnunk. pl.

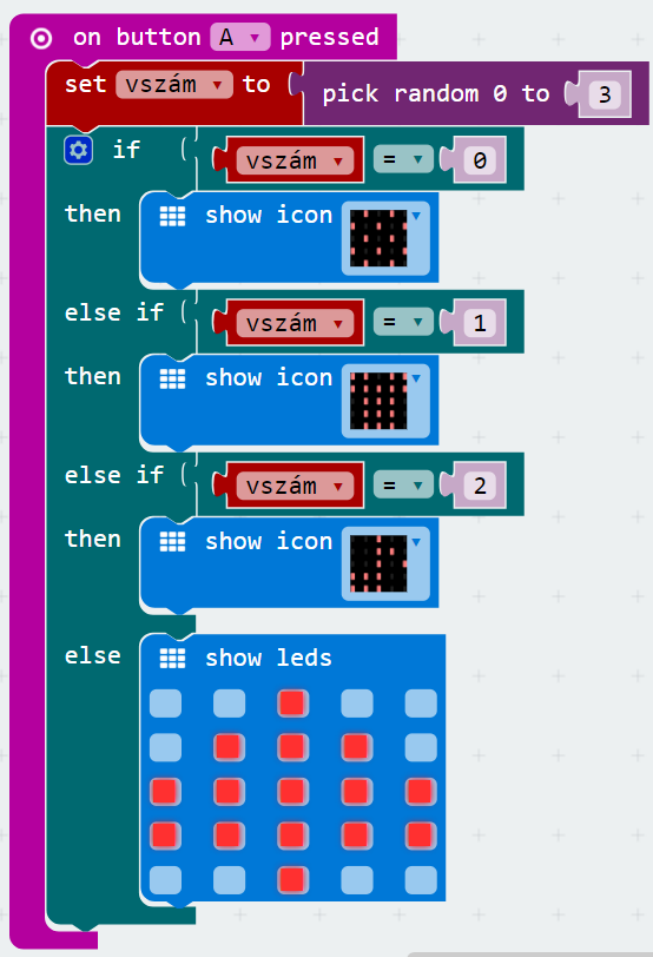
Ikon	Tevékenység	Ikon	Tevékenység
	Sakktáblát látunk? Sakkozzunk, vagy társasjátékozzunk barátainkkal.		Ha kotta jelet látunk, hallgassunk zenét, vagy menjünk koncertre.
	A póló a csapat sportokra utal. Menjünk focizni, vagy röpizni.		Ha egy fát látunk, menjünk kirándulni!

Generáljunk véletlenszámot akkora intervallumból, ahány program lehetőségünk van, jelen esetben négyet. (0-tól 3-ig).

Az elágazás behúzásakor a fogaskerék ikonra kattintva a ha, akkor különben elágazásból készíthetünk akár ha, akkor, különben ha akkor… elágazást is. Ehhez a szürke területről át kell húznunk a megfelelő elágazás elemeket a fehér területre.



Jelen esetben az alábbi struktúrát kell megvalósítanunk:



8. Kő, papír, olló játék

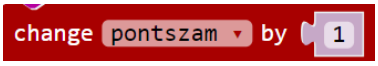
Feladat a diákok számára
Alakítsd át az előbbi programot úgy, hogy az eszköz megrázásakor véletlenszerűen egy kő, papír, vagy olló ikon jelenjen meg a kijelzőn!

9. Kő, papír, olló játék – pontok nyilvántartása

A tanár által elkészítendő/bemutatandő mintaprojekt
https://makecode.microbit.org/_67yJJWa2heV8

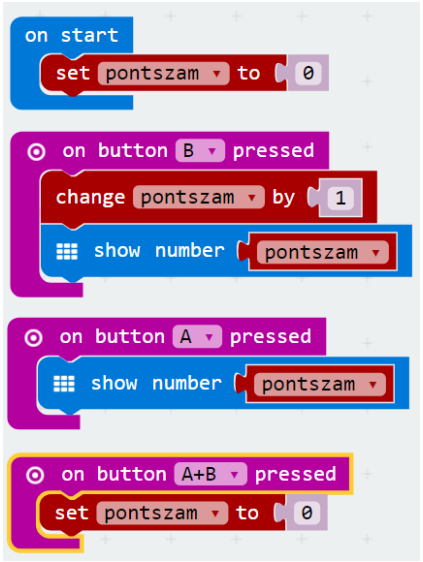
A diákok által kifejlesztett projektet fejlesszük tovább úgy, hogy a micro:bittel nyilván tudjuk tartani, hogy hány játékok nyertünk. A „B” gomb megnyomásával növelhessük meg a pontszámunkat, az „A” gombbal pedig kérdezhessük le az aktuális pontszámunkat. Az A és B gomb együttes lenyomásával a pontszám nullázódjon.

Mutassuk meg, hogy a változó értékét hogyan növelhetjük (csökkenthetjük).

Hívjuk fel a figyelmet arra, hogy a 

blokk használatával egyenértékű a következő megoldás is:

A megoldás:



Hagyjunk időt arra, hogy a diákok egymás ellen játszhassanak kő, papír, olló játékot. Minden nyert játszma után növeljék a pontszámukat. Vezényeljük le a játékot úgy, hogy minden diák azonos számú játékot játsszon. A végén mindenki mutassa meg, hogy mekkora pontszámot ért el, határozzuk meg, ki a győztes.

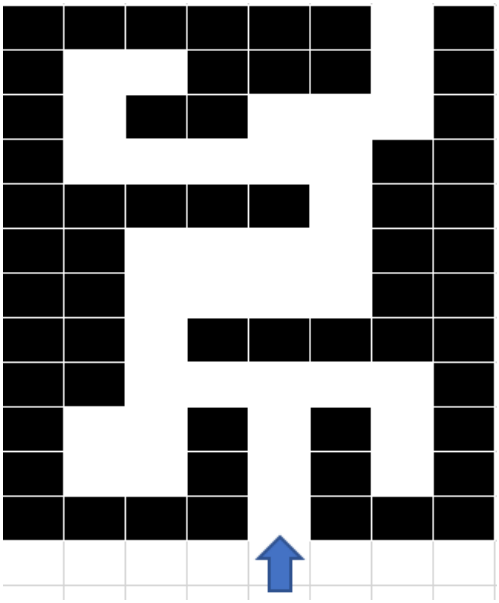
5. alkalom Labirintus játék: while ciklus, logika feltételek, tagadás, függvények használata

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Ismétlés (kijutás a labirintusból)	Önállóan megoldandó feladat a diákok számára a korábban bemutatott funkciók felhasználásával. Hasonlítsuk össze, hogy ki, mennyi lépésből tudott kijutni a labirintusból!	15 perc
2. Készítsünk labirintust	Tanári bemutató, új funkciók megismerése (adott koordinátájú pont állapotának lekérdezése, while ciklus használata), ötletelés.	10 perc
3. Menjünk végig a labirintuson (döntés jobbra)	Tanári bemutató, új funkciók megismerése (tagadás logikai műveletek).	15 perc
4. Menjünk végig a labirintuson (döntés 4 irányba)	A diákok továbbfejlesztik az alkalmazást úgy, hogy a pontot végig lehessen vezetni a labirintusban az eszköz 4 irányba történő döntésével. Nézzük meg a diákok megoldásait, beszéljük át, hogy milyen feltételeket kellett megadni.	20 perc
5. Labirintus, eredmény kijelzés	Tanári bemutató, új funkciók megismerése (függvény használata).	10 perc
6. A projekt továbbfejlesztése párokban	Hogyan lehetne továbbfejleszteni a játékot? Ötletelés párokban. Az alkalmazás továbbfejlesztése.	20 perc

1. Ismétlés (kijutás a labirintusból)

A korábban elhangzottak felelevenítéséhez adjuk ki az alábbi feladatot, melynek megoldásához már mindent ismernek a diákok.

Feladat a diákok számára



A képen látható labirintusból kell kijutnod, de csak a micro:bitet használhatod arra, hogy eldönts, 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 kijelzőre ez a szám. Az „A+B” gomb együttes megnyomásakor nullázodjon 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 haladnod, 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 fordulást. Ha sem jobbra, sem balra nem lehet tovább menni, akkor vége a játéknak, csapdába kerültél. Ilyenkor kezd újra a játékot a próbálkozások nullázásával. Ha sikerült kijutnod a labirintusból, nyomd meg a „B” gombot, hogy megtudd az irányváltások számát!

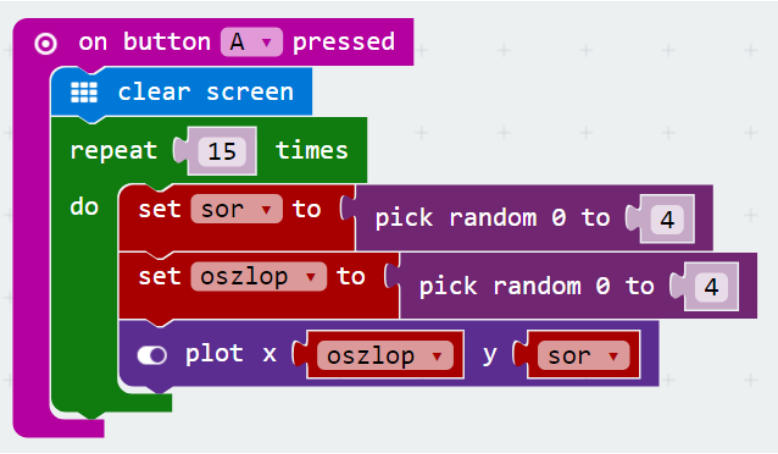
2. Készítsünk labirintust

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_iVxhvT8ejd1g

Következő példánkban a micro:bit segítségével fogunk előállítani véletlenszerűen labirintust, mégpedig úgy, hogy minden előállított labirintus pontosan ugyanannyi (pl. 15) darab kigyújtott pontból álljon.

Először induljunk ki az alábbi állapotból. Az „A” gomb hatására 15 alkalommal véletlen koordinátát generálunk és kigyújtjuk az ahhoz tartozó pontot.



Teszteljük le a megoldást. Kérdezzük meg a diákokat, hogy valójában hány pont jelent meg a kijelzőjükön. Látszik, hogy általában nem mind a 15 pont lesz kigyújtva, mert véletlenszerűen kiválaszthatunk olyan pontokat is, amelyek már ki vannak gyújtva.

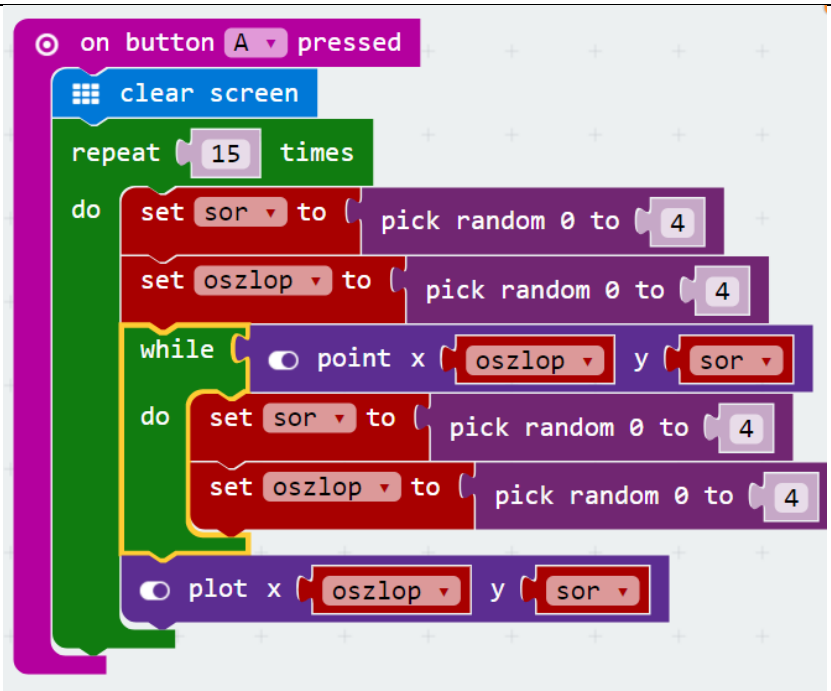
A megoldás az lehetne, hogy addig kellene új véletlenszámot generálni, míg üres képpontot nem találunk. Ezzel garantálhatnánk, hogy pontosan 15 pont legyen kigyújtva.

Azt, hogy egy adott pont ki van-e gyújtva a **Led** kategória **points x y** blokkjával kérdezhetjük le. Igaz (true) értéket kapunk vissza, ha ki van gyújtva a pont, és hamisat (false), ha nem.

Ötleteljünk, hogyan tudnánk ez alapján megvalósítani, hogy pontosan 15 pont legyen kigyújtva véletlenszerűen?

Megoldás lehet, hogy addig generálunk újabb és újabb koordinátákat véletlenszerűen, míg nem kapunk olyan pontot, ami nincs kigyújtva.

Ezt a **Loops** / **While *feltétel* do** ciklusával tudjuk megtenni az alábbi módon.



Így már ténylegesen annyi pont lesz kigyújtva, amennyit a `repeat` ciklusban beállítottunk.

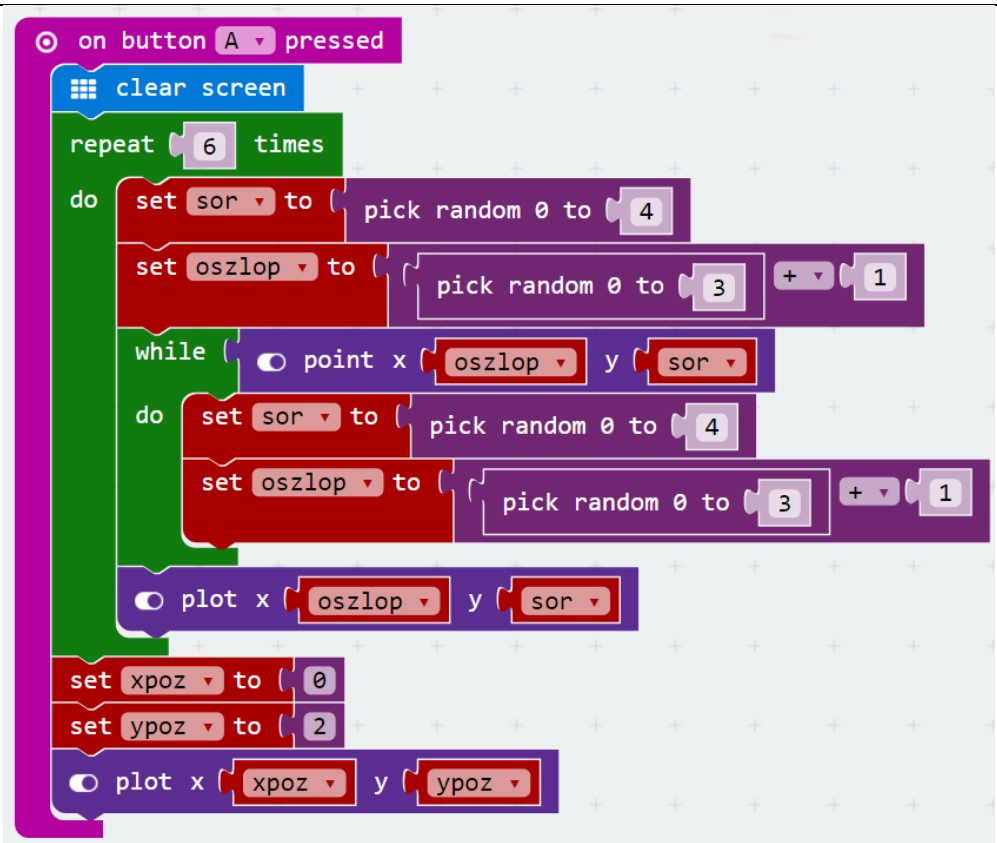
Próbáljuk ki a programot 24 ponttal is. Ilyenkor csak 1 pont marad, ami nem lesz kigyújtva.

3. Menjünk végig a labirintuson

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_1fPUwwfpMdsA

Most már tudunk véletlenszerűen labirintus létrehozni. Készítsünk olyan alkalmazást, amelyben végig tudunk vinni egy pontot a labirintuson. Az eszköz 4 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 labirintus generáljunk úgy, hogy az első oszlopba ne tegyünk akadályt.

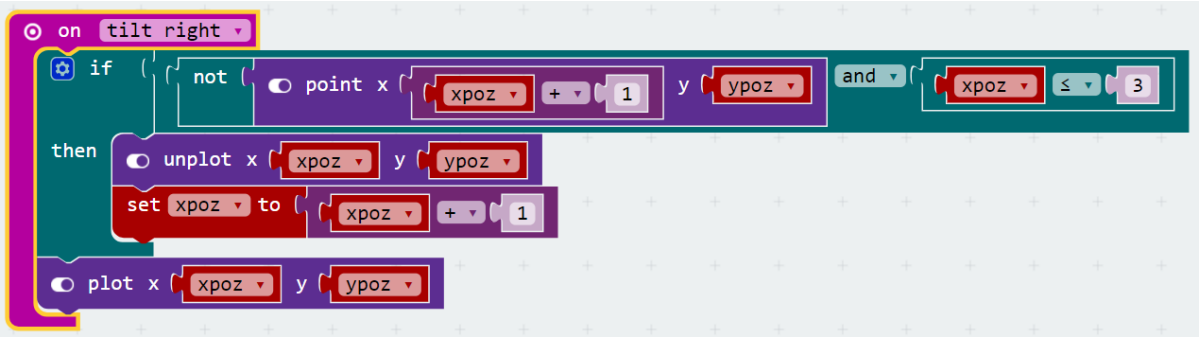


Beszéljük át közösen, hogy minek kell történnie akkor, ha jobbra döntjük az eszközt!

Egyrészt vizsgálnunk kell, hogy a jobb oldali szomszédos mező üres-e. Ha igen, akkor az eredeti helyéről törölnünk kell a pontot, és az új helyen meg kell jelenítenünk azt, és ezt a koordinátát el kell tárolnunk. Emellett azt is néznünk kell, hogy legfeljebb az utolsó előtti oszlopban lehetünk a jobbra lépés előtt, mivel nem szabad kilépnünk a területből (xpoz<=3).

Az üres hely vizsgálatánál használjuk a `not` blokkot, így a feltételünk akkor lesz igaz, ha üres az adott pont.

A megoldás tehát:



Most a diákokon a sor, hogy mind a 4 irányú döntésre elkészítsék a megfelelő kódot.

4. Menjünk végig a labirintuson (döntés 4 irányba)

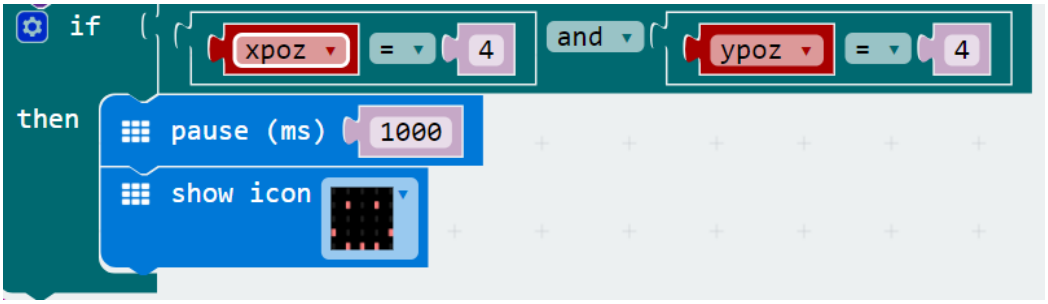
Feladat a diákok számára
Alakítsd át az előbbi programot úgy, hogy a pontot az eszköz 4 irányban történő döntésével `(tilt right, title left, logo up, logo down)` lehessen irányítani.

5. Labirintus, eredmény kijelzés

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_Tcxb2Riuf776

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!

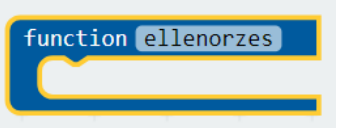
Ezt megoldhatjuk úgy, hogy az alábbi blokkot elhelyezzük mind a 4 blokkunk végén.

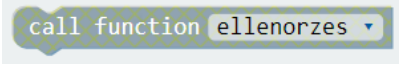


De ha ezt tesszük, és úgy döntünk, hogy például ne vigyorgó fej jelenjen meg, hanem egy animáció, esetleg egy szöveg, akkor innentől 4 helyen kell módosítani a kódunkat. Ez nagyon nem hatékony. Ezért azon kódrészleteket, amelyek többször is felhasználásra kerülnek érdemes függvényekbe szervezni.

Ehhez a **functions** kategória **make a function** gombját kell megnyomnunk és meg kell adnunk a függvény nevét. A név legyen a következő: ellenorzes.

Ekkor megjelenik a következő blokk:



Ebben elhelyezhetjük a megfelelő kódot. A függvényt a 

blokkal hívhatjuk meg, amely automatikusan létrejön a függvény létrehozásakor.

Mind a 4 blokk végére helyezzük el a függvény hívást.

A teljes projektünk elérhető a https://makecode.microbit.org/_Tcxb2Riuf776 címen.

Megjegyzés: mivel háttérfolyamat használatára is lehetőségünk van a `forever` *blokk használatával, azt is megtehetjük, hogy a* `forever` *blokkban helyezzük el az „ellenorzes” függvényhívást. Ekkor a többi blokkból törölhetjük is ezt a függvényhívást.*

6. A projekt továbbfejlesztése párokban

Feladat a diákok számára

Alakítsatok párokat, és beszéljétek meg, hogy hogyan tudnátok továbbfejleszteni a labirintus játékot, hogy a játékélmény jobb legyen. Kezdjétek el az ötlet megvalósítását.

Tanárként mi magunk is adhatunk ötleteket a továbbfejlesztésre, ha a pároknak nem sikerül megfelelő nehézségű továbbfejlesztési ötlettel előállni. Például:

- Minden lépést kövesse hanghatás, a végén pedig játszódjon le dallam
- Ha sikerült kijutni a labirintusból, akkor a következő indításnál már nehezebb legyen a pálya, mindig eggyel több akadály legyen kirajzolva.
- Vezessük be a teleportálás funkciót. Ha a középső pont nincs kigyújtva, akkor lehessen oda ugrani az eszköz megrázásával.

6. alkalom Haladó játékfejlesztés spriteokkal

A spriteok (magyarul manók) segítségével még egyszerűbben készíthetünk játékokat.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Bevezető haladóbb játékok készítéséhez (Űrhajós játék)	Tanári bemutató, új funkciók megismerése (sprite létrehozása, mozgatása).	10 perc
2. Űrhajós játék folytatása	Tanári bemutató, új funkciók megismerése (ütközés kezelése, pontszám beállítása, pontszám növelése, játék befejezése). A játék kipróbálása.	20 perc
3. A projekt továbbfejlesztése egyénileg	A diákok továbbfejlesztik az alkalmazást a kiadott feladatnak megfelelően. Nézzük meg a diákok megoldásait, beszéljük át, hogy milyen feltételeket kellett megadni.	15 perc
4. Ötletelés, játékfejlesztés párokban	Hogyan lehetne továbbfejleszteni a játékot, vagy más játékot készíteni a sprite-ok felhasználásával? Ötletelés párokban. Az alkalmazás továbbfejlesztése.	35 perc
5. Az elkészült alkalmazások áttekintése, bemutatása	Az elkészült munkákat nézzük meg közösen!	10 perc

1. Bevezető haladóbb játékok készítéséhez (Űrhajós játék)

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

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_f50HbUTyAU49

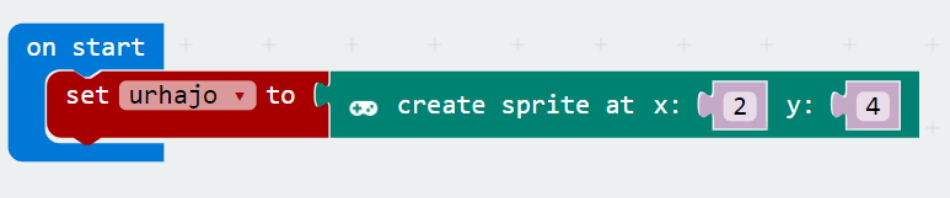
Készítsünk egy játékot. A játék lényege: A kijelző alsó sorában jelenjen meg egy pont. Ez lesz az űrhajó. Az „A” és „B” gombbal lehessen balra és jobbra mozgatni. Fentről jöjjön meteor véletlenszerű helyről, amelyet ki kell kerülni. Amikor a meteor elérte az alsó sort, kerüljön a felső sorba véletlenszerű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

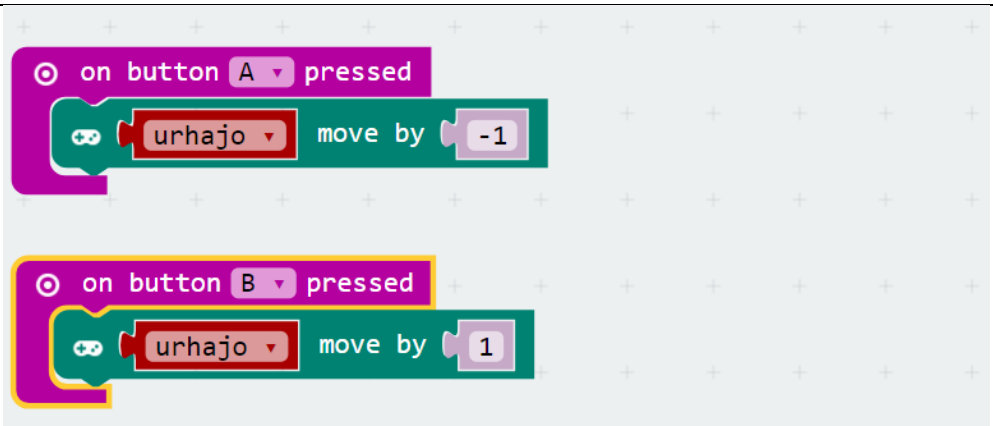
Először ismerkedjünk meg azzal, hogy hogyan mozgathatunk egy pontot a legegyszerűbben a képernyőn. Ehhez úgynevezett spritetot (rajzobjektumot) kell használnunk. A sprite olyan objektum, aminek van iránya, tud előre és hátra lépni, lekérdezhetjük, hogy ütközött-e más rajzobjektumokkal, vissza tud pattanni a falról, és így tovább.

Először hozzuk létre közösen az alábbi projektet.

A megfelelő blokkokat a **Game** kategóriában találhatjuk, amely a **Haladó (Advanced)** lehetőségek között helyezkedik el.

- Az űrhajó részére hozzunk létre egy spriteot, amelynek kezdő koordinátája legyen az X=2, Y=4, vagyis alul, középen legyen elhelyezve.
- Ezután beállíthatjuk, hogy az „A” gomb megnyomására lépjen hátra egyet az űrhajó (mivel jobbra néz, ezért ez igazából azt jelenti, hogy balra lép egyet).
- A „B” gomb hatására pedig előre (jobb oldali irányba) kell mennie.





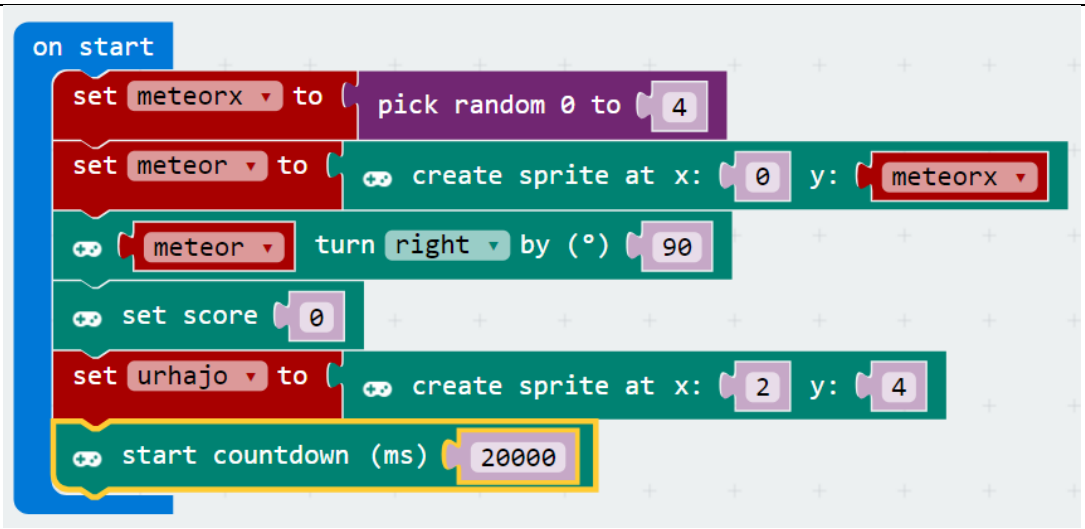
Láthatjuk, hogy az A és B gombok hatására az űrhajó a szomszédos helyekre megy. Látszik, hogy nem tudunk kimenni a kijelzőről, vagyis nincs szükség olyan pozíció ellenőrzésre, mint amit korábban végeztünk.

2. Űrhajós játék (folytatása)

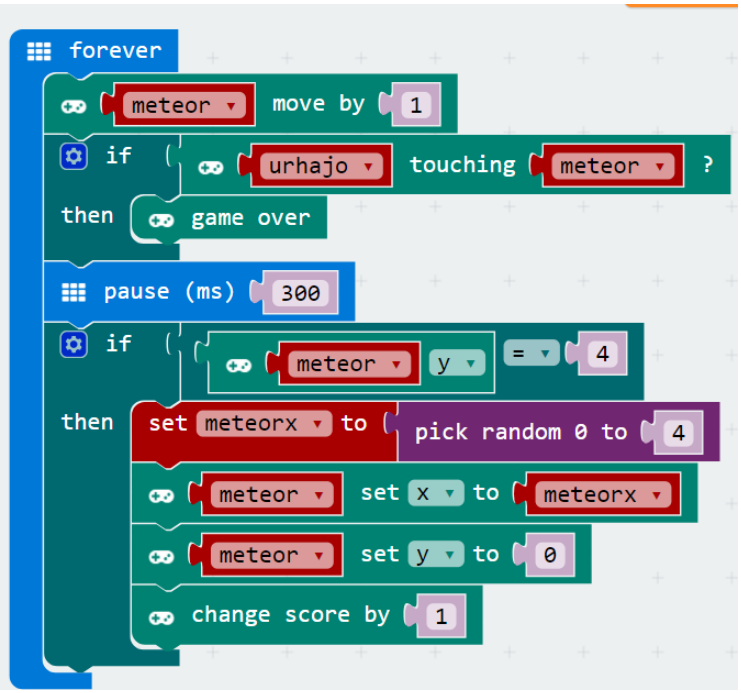
A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_HkP1XFhXTJTX

Folytassuk az alkalmazás fejlesztését közösen.

- A játék kezdetekor válasszunk egy véletlenszerűen meghatározott x koordinátát a meteor számára (meteorx).
- Majd a meteor számára hozzunk létre egy spriteot.
- Mivel létrehozásakor a sprite jobbra néz, el kell fordulnunk vele 90 fokot, hogy lefele nézzen, hiszen felülről lefele kell majd esnie.
- Állítsuk be a kezdeti pontszámot (score) nullára.
- Az űrhajó részére már létrehoztuk a spriteot.
- Indítsunk el egy számlálót (counter), hogy legfeljebb 20 másodpercig tartson a játék. 20 másodperc elteltével automatikusan megjelenik az elért pontszám.



- Ezek után egy háttérfolyamatot kell indítanunk a **forever** blokkban.
- Beállítjuk, hogy a meteor lépjen előre 1-et.
- 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 alsó sort (y=4), akkor majd újra meg kell jelennie a felső sorban. Válasszunk egy véletlenszerűen választott x koordinátát, és helyezzük el a meteort az első sorban (Y=0).
- Növeljük meg a pontszámot 1-el.



3. A projekt továbbfejlesztése egyénileg

Feladat a diákok számára
– Legyen az űrhajónak pajzsa. Ez jelentse azt, hogy pl. 2 ütközést kibír az űrhajó, és csak az utána lévő ütközéskor ér véget a játék! – Az űrhajót ne csak az „A” és „B” gombbal lehessen irányítani, hanem az eszköz balra, illetve jobbra döntésével is! – Adj hangeffektet a projekthez!

4. Ötletelés, játékfejlesztés párokban

Feladat a diákok számára
Hogyan lehetne továbbfejleszteni a játékot, hogy növekedjen a játékélmény? Ötleteljetek a játékkal kapcsolatban és valósítsátok meg az elképzeléseiteket!

Mi magunk is adhatunk ötleteket a továbbfejlesztésre, ha a pároknak nem sikerül megfelelő nehézségű továbbfejlesztési ötlettel előállni. például

– Egyszerre több meteor is jöjjön az űrhajóval szemben, azok között kelljen navigálni.

– Legyen olyan elem, amit nem kikerülni kell, hanem elkapni, ilyenkor a pajzsunk töltődjön fel!

– A pajzs értéke a legalsó sorban látszódjon. Minél több pont van kigyújtva, annál nagyobb a pajzsunk ereje. Az ütközéskor csökkenjen le a pajzsunk ereje.

5. Az elkészült alkalmazások áttekintése, bemutatása

A csoportok nézzék meg és próbálják ki egymás munkáit!

7. alkalom Haladó játékfejlesztés spriteokkal (2. rész)

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Tenisz meccs	Tanári bemutató, komplex játék fejlesztésének lépései.	30 perc
2. Tenisz meccs továbbfejlesztése párokban	A diákok pármunkában továbbfejlesztik az alkalmazást, majd bemutatják egymásnak a munkáikat és kipróbálják azokat.	30 perc
3. Ötletelés, játékfejlesztés párokban	Hogyan lehetne más játékot készíteni a sprite-ok felhasználásával? Ötletelés párokban.	15 perc
4. Az ötletek bemutatása	Az elkészült terveket beszéljük át közösen!	15 perc

1. Tenisz meccs

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_iH1HxtJy6ACM

Készítsünk egy komplexebb játékot lépésről lépésre. Ennek során felhasználjuk a spriteokban rejlő lehetőségeket, valamint gyakoroljuk a korábban tanultakat.

A játék lényege:

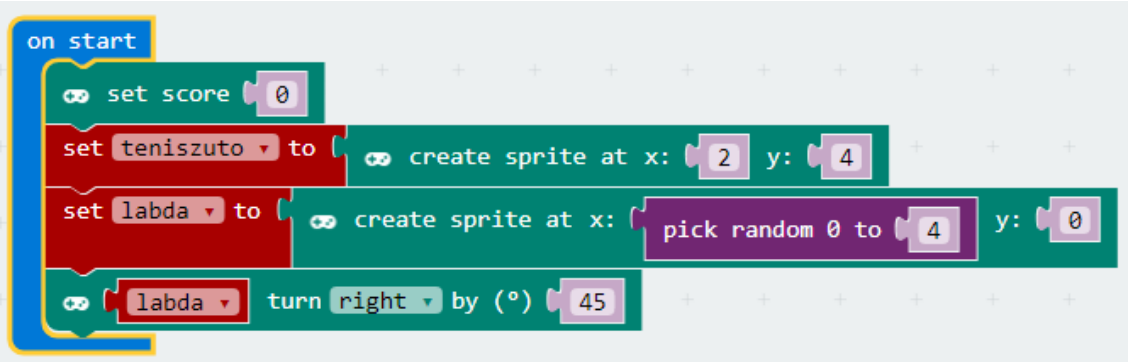
Egy tenisz meccset szimulálunk. A LED kijelző felső sorából indul a labda, amelyet vissza kell ütnünk a legalsó sorban elhelyezett teniszütővel. A teniszütő balra és jobbra tudjon mozogni. A mozgatást az A és B gombbal, valamint az eszköz balra és jobbra döntésével lehessen elvégezni.

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

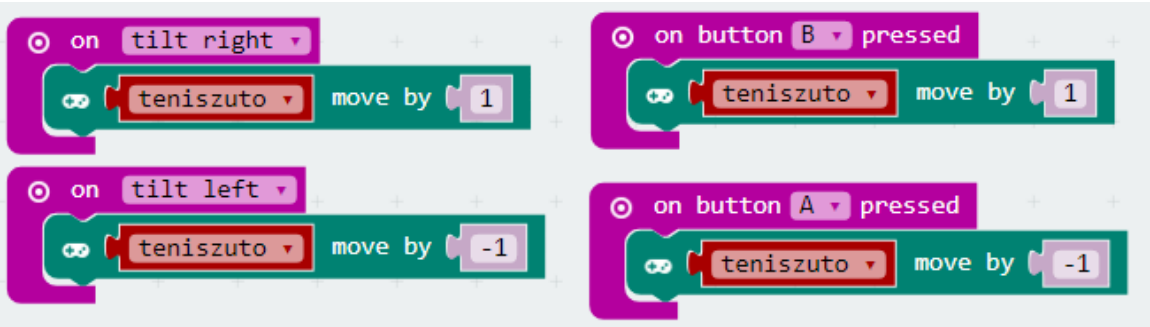
A játék kezdetekor állítsuk be a pontszámot nullára. Készítsünk *teniszuto* néven egy spriteot és helyezzük el a legalsó sorban, középen. Hozzuk létre a labdát is, a legelső

sorban, de véletlenszerűen választott helyen. Mivel a sprite alapesetben jobbra néz, fordítsuk el jobbra 45 fokkal, 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.



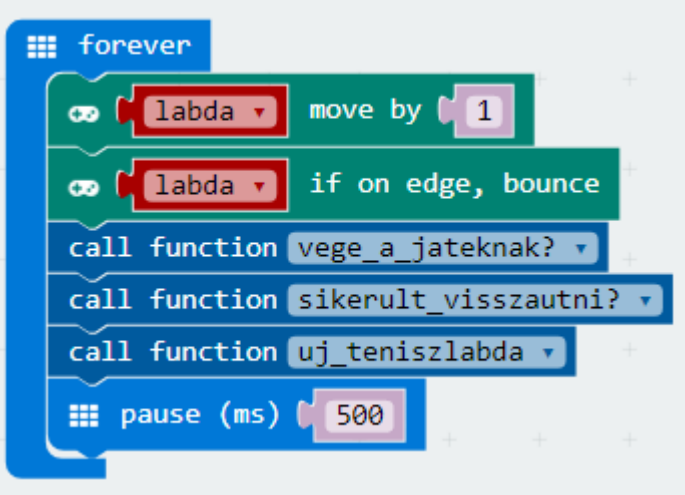
A játék motorja

A `forever` blokkban helyezhetjük el a játék folyamatos működtetéséért felelős blokkot. A játék során a következőket kell tennünk:

- 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 az a legalsó sort (y=4 koordináta), viszont a labda és az ütő X koordinátája nem egyezik meg, vagyis különböző helyen van a labda és az ütő. Egyelőre csak egy függvényhívást helyezzünk el itt *„vege_a_jateknak?”* néven, később megfogalmazzuk a tartalmát.
- Szintén ellenőriznünk kell, hogy sikerült-e visszaütni a labdát. Ez akkor következik be, ha az ütő érinti a labdát. Ekkor a pontszámot meg kell 1-el növelnünk. Ezt az ellenőrzést írjuk majd a *„sikerult_visszautni?”* 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

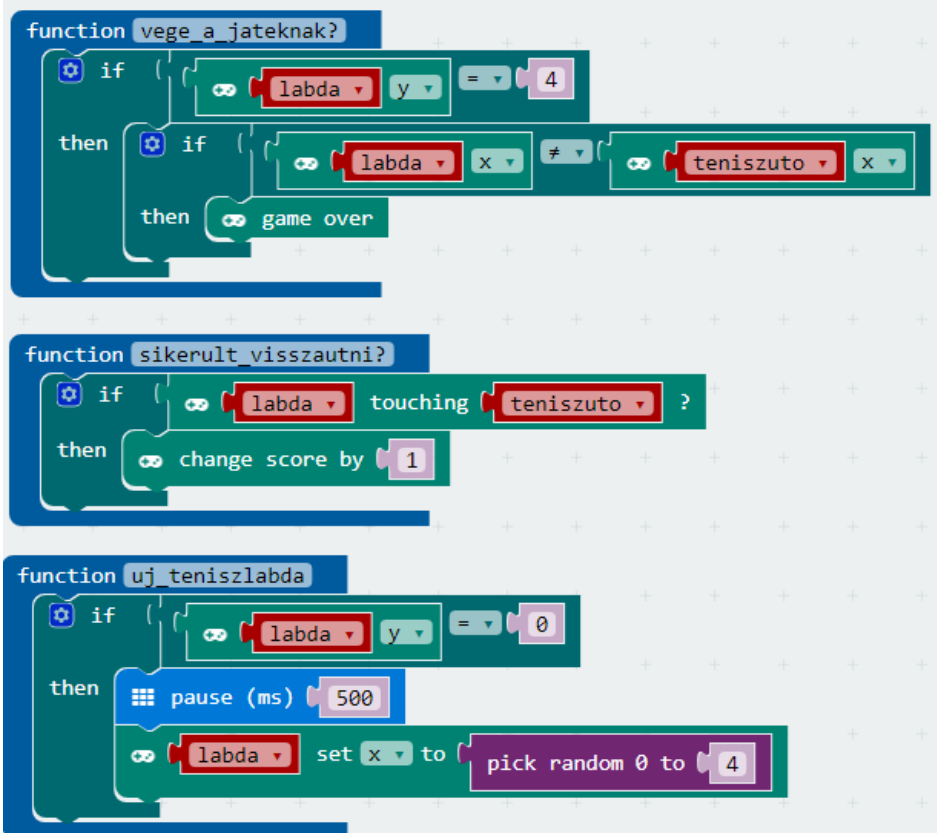
hiányában könnyen előfordulhatna, hogy a labda újra és újra ugyanazt az útvonalat járja be. Ezt a kódot fogjuk az „*uj_teniszlabda*” függvénybe írni.

- A blokkunk végén érdemes elhelyezni egy kis késleltetést, különben túl gyors lenne a játék.
- Vagyis a **forever** blokkunk most így néz ki:



A függvények megírása:

Nem maradt más hátra, minthogy megírjuk a megfelelő függvényeket!



Hagyjunk időt arra, hogy a diákok kipróbálhassák a játékot, és összehasonlíthassák, hogy mekkora eredményt értek el.

2. Tenisz meccs továbbfejlesztése párokban

Feladat a diákok számára

Hogyan lehetne továbbfejleszteni a játékot, hogy növekedjen a játékélmény? Ötleteljetek párokban, és fejlesszétek tovább az alkalmazást párokban, vagy egyénileg!

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

Mi magunk is adhatunk ötleteket a továbbfejlesztésre, ha a pároknak nem sikerül megfelelő nehézségű továbbfejlesztési ötlettel előállni. Például:

- Kezdetben lassabb legyen a játék, de ahogy egyre több labdát sikerült visszaütni, gyorsuljon fel a játék.
- Legyenek hanghatások a játék közben!
- Ha elértünk legalább 3 pontot és hibázunk egyet, ne érjen véget a játék, menjen tovább, de a pontszámunk csökkenjen eggyel.
- Amennyiben úgy sikerül visszaütni a labdát, hogy az vagy a bal felső sarokba, vagy a jobb felső sarokba kerül, akkor kapjunk extra 1 pontot!
- Kezdetben legyen szélesebb a teniszütőnk (mintha két ütő lenne egymáshoz ragasztva), és a játék előrehaladtával csökkenjen le a mérete.
- Az A+B gombok együttes megnyomásával kérhessünk új labdát. Ekkor a legfelső sorba kerüljön vissza a labda. Ezt a segítséget csak egyszer vehessük igénybe a játék során.

3. Ötletelés, játékfejlesztés párokban

Feladat a diákok számára

Most már sokféle funkciót ismersz a játékok készítéséhez. Milyen más játékokat lenne jó megvalósítani? Ötleteljetek párokban! Írjátok le a játékotok lényegét, mi jelenne meg a kijelzőn, mi a játék lényege, hogyan lehet irányítani az egyes objektumukat, stb.

Következő alkalommal meg is valósíthatjátok az ötleteiteket.

4. Ötletek bemutatása

A diákok mutassák be ötleteiket. Amennyiben a játék túl komplex, hívjuk fel a figyelmet az egyszerűsítési lehetőségekre. Amennyiben túl egyszerű, adjunk ötleteket a továbbfejlesztésekhez!

8. alkalom Saját játékok megvalósítása

Ezen alkalommal minden diák (vagy pár) megvalósíthatja a múlt alkalommal megtervezett alkalmazását.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Játékok fejlesztése	A diákok elkészítik a tervek alapján a játékokat. A tanár körbejár, segíti a munkát, válaszol a felmerülő kérdésekre, ötleteket ad a fejlesztésre vonatkozóan.	70 perc
2. Játékok bemutatása	Minden diák bemutatja az elkészült játékát. A diákok kipróbálják egymás játékait. A tanár és a diákok értékelik egymás munkáit (mi volt a jó a játékban, hogyan lehetne továbbfejleszteni, ...)	20 perc

9. alkalom Ismerkedés a szenzorokkal

A micro:bit számos szenzort tartalmaz. Ebben a foglalkozásban ezekkel ismerkedünk meg.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Fényérzékelő használata	Tanári bemutató, új funkciók megismerése (fényérzékelő).	5 perc
2. Fény és hang (önálló munka)	A diákok továbbfejlesztik az alkalmazást a kiadott feladatnak megfelelően. Ezután megnézzük (meghallgatjuk) egymás megoldásait.	15 perc
3. Hőmérséklet érzékelő, hang jelzéssel	Tanári bemutató, új funkciók megismerése (hőmérséklet érzékelés).	15 perc
4. Hőmérséklet kijelzés egyéni grafikonnal (önálló munka)	A diákok továbbfejlesztik az alkalmazást a kiadott feladatnak megfelelően. Ezután megnézzük egymás megoldásait.	20 perc
5. Iránytű használata	Tanári bemutató, új funkciók megismerése (iránytű).	5 perc
6. Iránytű készítése (önálló munka)	A diákok továbbfejlesztik az alkalmazást a kiadott feladatnak megfelelően. Ezután megnézzük egymás megoldásait.	15 perc
7. Gyorsulásmérő bemutatása	Tanári bemutató, új funkciók megismerése (gyorsulásmérő használata).	10 perc
8. Ötletelés	Ötleteljünk közösen arról, hogy például játékok készítése során hogyan tudnánk felhasználni a szenzorokban rejlő lehetőségeket. Jegyezzük fel az ötleteket, későbbi megvalósításhoz.	5 perc

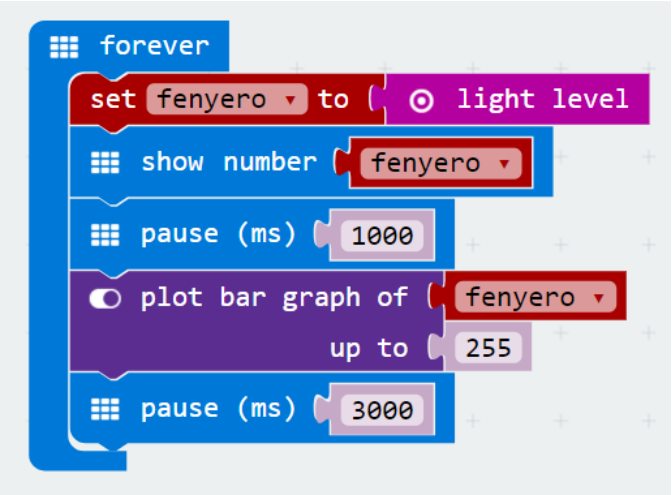
1. Fényérzékelő használata

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_106TFm3sce1P

A micro:bit ledjei nem csak kijelzőként működnek, hanem fényérzékelőként is. Az **input** kategóriában a **light level** blokk használatával tudjuk lekérdezni a fényerősség szintjét. Ez 0 és 255 között mozog attól függően, hogy teljes sötétséget, vagy erős fényt érzékel az eszköz.

De hogy tudjuk ezt szemléltetni? Például úgy, hogy a fényerősség értékét megjelenítjük számként, illetve egy grafikonon is ábrázolhatjuk. A **plot bar graph** blokk első paraméterébe egy konkrét számot, vagy változót írhatunk, a második paraméterébe pedig azt kell megadnunk, hogy mekkora lehet a maximális érték.

Valósítsuk meg az alábbi alkalmazást:



Takarjuk le a kezünkkel a micro:bitet, illetve világítsuk meg lámpával. Látni fogjuk, hogy a grafikon hogyan változik az eltérő fényviszonyok mellett.

2. Fény és hang

Feladat a diákok számára

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.

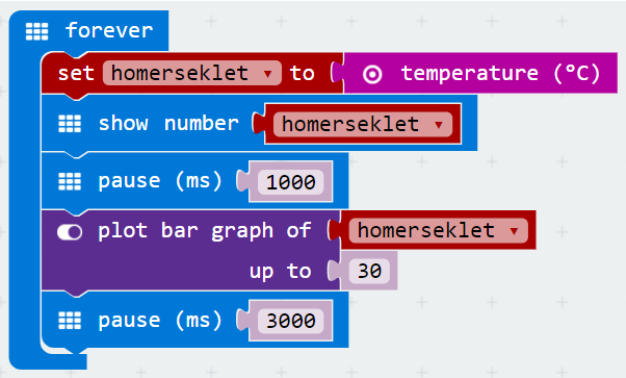
3. Hőmérséklet érzékelő, hang jelzéssel

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_JHd2ifYeUH8o

A micro:bit a hőmérsékletet is mérni tudja. Az **input** kategóriában a **temperature** blokk használatával tudjuk lekérdezni a hőmérsékleti értéket.

Alakítsuk át a korábbi alkalmazásunkat úgy, hogy a hőmérsékleti érték és annak grafikonja váltakozzon a kijelzőn! Állítsuk be a maximális értéket 30-ra.



Készítsünk olyan eszközt, amelyet a vak emberek is használhatnának a hőmérséklet lekérdezésére. Ilyenkor hanggal tudnánk szemléltetni a hőmérsékleti értéket.

Ötleteljünk arról, hogy hogyan lehetne ezt megcsinálni?

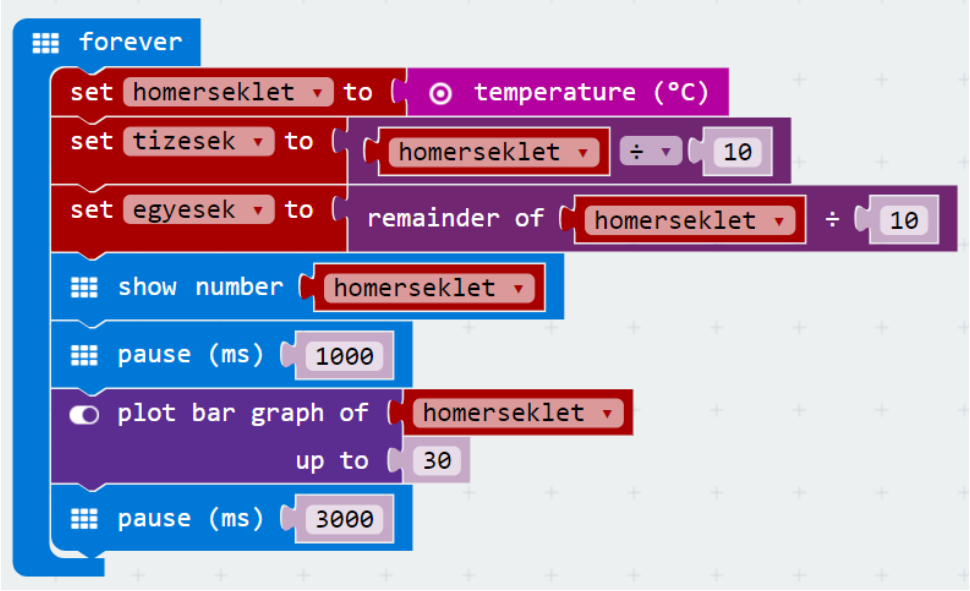
- Annyiszor lejátszunk egy hangot, ahány fok van? (hátrány, sokat kell számolni a vak embernek, sok ideig tart)
- Jobb lehet, ha külön játsszuk le a 10-es helyiértéken lévő számot, és külön az 1-es helyiértéken lévőt. Így a 23 fokot lejátszhatnánk úgy, hogy kettőt pittyeg az eszköz, majd kis szünet után, pl. magasabb pittyeg hármat.

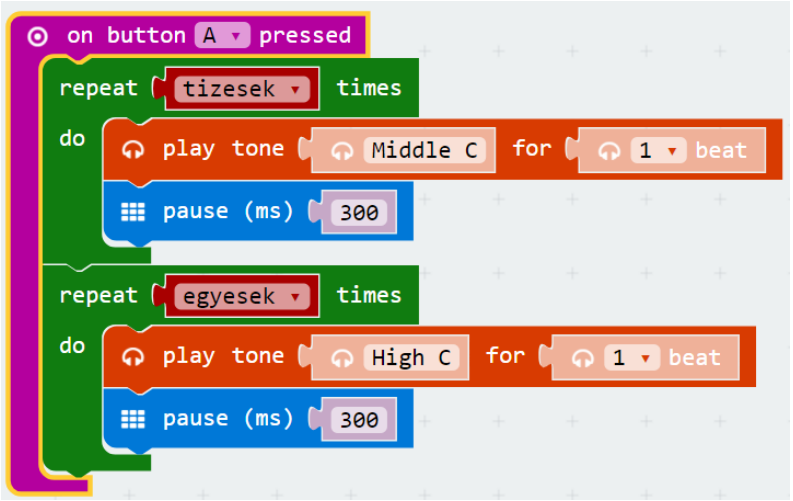
Valósítsuk meg ez utóbbit! Ehhez meg kell határoznunk, hogy az adott hőmérsékleti értékben hányszor van meg maradék nélkül a 10, illetve a maradékot is meg kell határoznunk.

Készítsünk ehhez változókat, és használjuk a **math** kategória megfelelő **blokkjait** (**osztás**, **maradék [remainder of]**)

Az „A” gomb megnyomásakor játsszuk le a megfelelő számú hangot.

A megoldás:





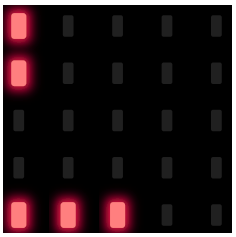
4. Hőmérsékleti grafikon

Feladat a diákok számára

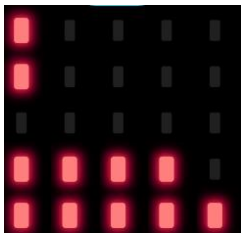
A beépített grafikon leolvasása eléggé nehézkes. Készíts olyan grafikont, amely a következőképpen ábrázolja a számokat.

A felső két sorban ábrázoljuk a tizes helyiértéken lévő számokat, két pont magas pálcikákkal. Az alsó két sorban pedig az egyes helyiértéknek megfelelő számú pontot gyújtsunk ki.

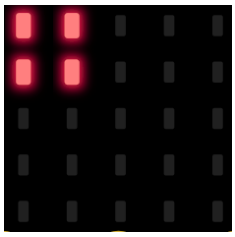
Néhány példa:



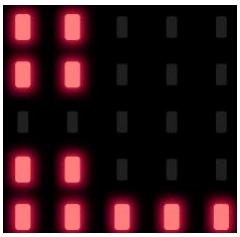
13°



19°



20°



27°

Ez a grafikon akkor jelenjen meg, amikor a „B” gombot nyomjuk le.

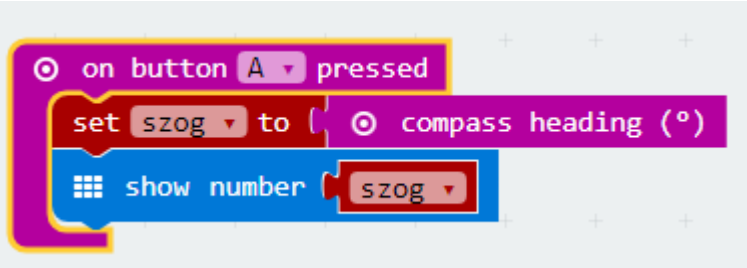
5. Iránytű használata

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_L6uTq339pT8k

A micro:bit egy iránytűt is tartalmaz. Az **Input** kategóriában a **compass heading** blokk használatával tudjuk lekérdezni az iránytű által mutatott szög értéket.

Magyarázzuk el a gyerekeknek, hogy mit jelentenek a különböző értékek. (0° = Észak, 90°= Kelet, 180° = Dél, 270° = Nyugat). Az alábbi iránytűt ki is vetíthetjük (http://bit.ly/2GJk6cK).

Készítsük el közösen ezt az egyszerű alkalmazást. Az „A” gomb megnyomásakor tároljuk el változóban a szög értékét (amely 0 és 359 fok között vehet fel értékét), majd írassuk ki a kijelzőre.



Az első futtatásnál azt tapasztaljuk, hogy megjelenik a *„Draw a circle” (Rajzolj kört)* szöveg a kijelzőn. Ennek oka, hogy az iránytű használata előtt kalibrálni kell az eszközt. Döntsük el az eszközt különböző irányokba úgy, hogy a kijelzőn egy körvonal jelenjen meg. Ezután már tesztelhetjük az alkalmazásunkat.

6. Iránytű készítése (önálló munka)

Feladat a diákok számára

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 46 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)

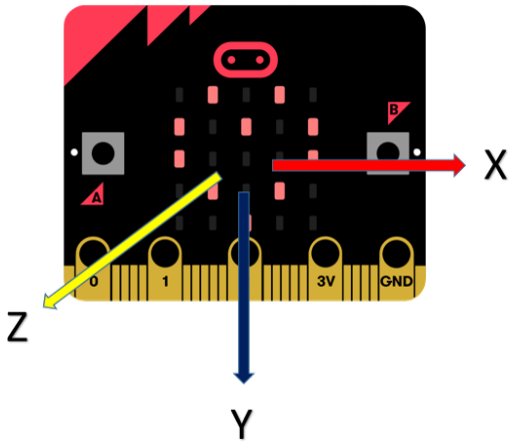
A nyilakat nem kell megrajzolni, a **Basic** kategória / **Show arrow** blokkjában ezek egyszerűen kiválaszthatóak.

7. Gyorsulásmérő használata

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_L1JhPrVWbKgw

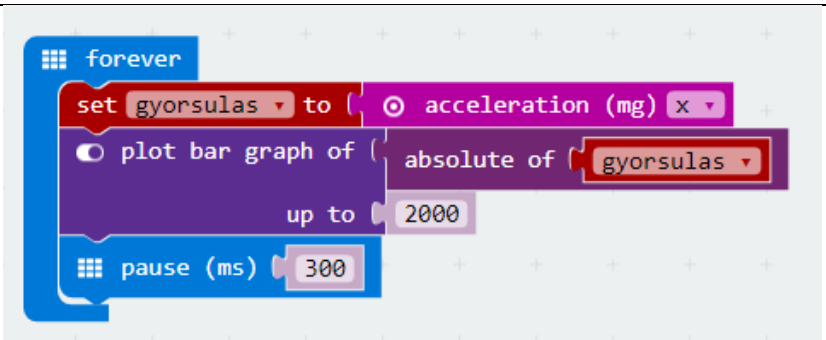
Az érzékelők között gyorsulásmérőt is találunk. A gyorsulásmérő három tengely mentén tud mérni. Az alábbi ábra szemlélteti az egyes tengelyeket.



A szabadon eső testek gyorsulása egyenletes, az úgynevezett nehézségi (gravitációs) gyorsulás, jele g. Normál értéke g = 9,80665 m/s² kerekítve 9,81 m/s² Fontos tudnunk, hogy a g nehézségi gyorsulás a földrajzi helyzettől és a felszíni magasságtól is függ.

A micro:bit nem a g értéket, hanem annak ezred részét adja vissza.

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

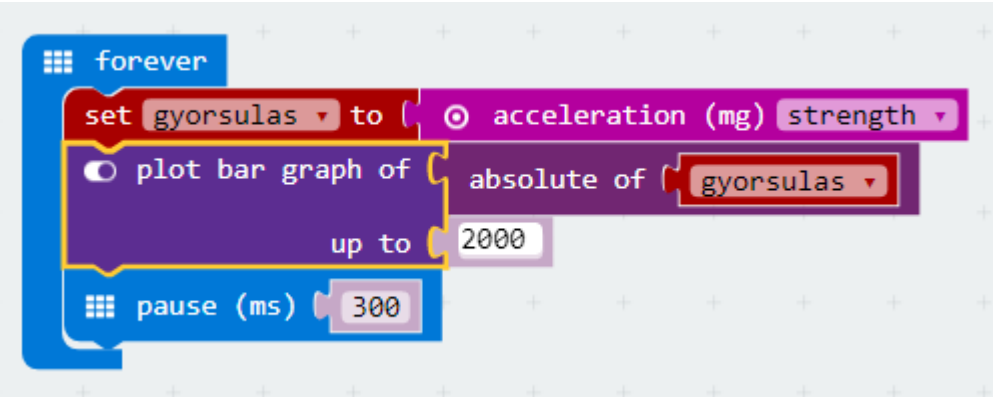


Töltsük le a kódot a micro:bitre. Azt tapasztaljuk, hogy az X irányú mozgás hatására történik változás a grafikonon, ha kizárólag(!!!) függőlegesen mozdítjuk el az eszközt, vagy csak magunk felé mozgatjuk, nem tapasztalunk változást.

Változtassuk meg a kódot úgy, hogy az y mentén történő gyorsulást jelezzük ki, és próbáljuk ki az alkalmazást a gyakorlatban is! Ha az ábrának megfelelően tartjuk az eszközt, akkor azt tapasztaljuk, hogy az y tengely mentén nyugalmi helyzetben is kb. 1000-es értéket (vagyis 1 g-t) mérünk (a grafikonon kb. a pontok fele lesz kigyújtva).

Változtassuk meg a kódot úgy, hogy az y mentén történő gyorsulást jelezzük ki, és próbáljuk ki az alkalmazást a gyakorlatban is! Ekkor azt tapasztaljuk, hogy akkor változik a grafikon, ha magunk felé, vagy tőlünk távolodóan mozgatjuk az eszközt.

Ha nem szeretnénk külön tengelyek mentén mérni a gyorsulást, hanem egy összesített eredményt szeretnénk, akkor válasszuk ki a strength (erősség) értéket!



Próbáljuk ki ezt a változatot is!

10. alkalom Játék a szenzorokkal

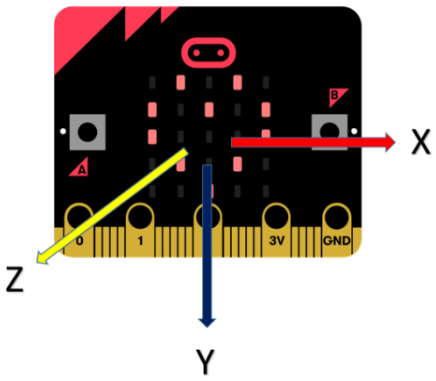
Múlt alkalommal megismerkedtünk a szenzorokkal, most haladóbb alkalmazásokat készítünk a szenzorok által mért adatok felhasználásával.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Gyorsulás kijelzés fényerősség módosítással	Ismétlő feladat a diákok számára. A megoldás beszéljük át közösen.	10 perc
2. Ugrás számláló	Tanári bemutató, új funkciók megismerése. Az alkalmazás kipróbálása, verseny a diákok között, 30 mp alatt ki éri el a legnagyobb pontszámot ugrálással?	10 perc
3. Ugrás számláló továbbfejlesztése (páros munka)	A diákok párokban ötletelnek és továbbfejlesztik az alkalmazást.	20 perc
4. Tojás és kanál játék	Tanári bemutató, az alkalmazás kipróbálása.	10 perc
5. Saját alkalmazás fejlesztése a szenzorok felhasználásával	A diákok párokban ötletelnek és elkészítik a saját alkalmazásaikat.	30 perc
6. Az elkészült munkák bemutatása	A diákok bemutatják és kipróbálják egymás alkalmazásait.	10 perc

1. Gyorsulás kijelzés fényerősség módosítással

Feladat a diákok számára

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



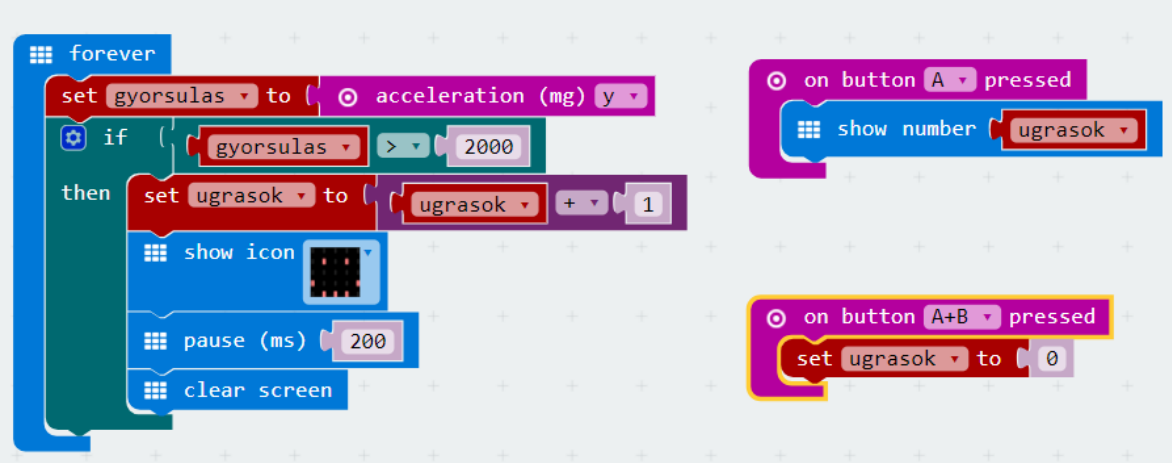
Indításkor minden Led legyen kigyújtva. A háttérben futó folyamatban egy változóban tárold el az x irányú gyorsulás abszolút értékét, a fényerőt pedig ezen érték ötödére állítsd be!

Próbáld ki az alkalmazást úgy, hogy oldalirányba mozgatod az eszközt!

2. Ugrás számláló

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_AMpMwfREUiqh

Most egy olyan alkalmazást készítünk, amely számolja az ugrásainkat. Ennél a feladatnál is a gyorsulásmérő által mért értékeket fogjuk felhasználni. (Az eredeti alkalmazás a https://makecode.microbit.org/lessons/pogo/activity címen található)



Készítsük el közösen a fenti alkalmazást. Ennek lényege, hogy folyamatosan vizsgáljuk az Y irányú gyorsulás értékét, és ha az meghaladja a 2000-es értéket, akkor eggyel növeljük az ugrások számát, és megjelenítünk egy vigyorgó fejet. Az „A” gombbal ki tudjuk íratni az ugrások számát, az „A+B” gombokkal pedig le tudjuk nullázni az ugrás értéket.

Próbáljuk ki az alkalmazást! Azt tapasztaljuk, hogy a kisebb ugrásoknál eggyel növekszik az ugrások száma, a nagyobb ugrásoknál viszont akár kettővel is.

Versenyeztessük meg a diákokat. Adjunk 30 másodpercet mindenkinek arra, hogy ugráljon, a végén hasonlítsuk össze a pontszámokat!

3. Ugrás számláló továbbfejlesztés

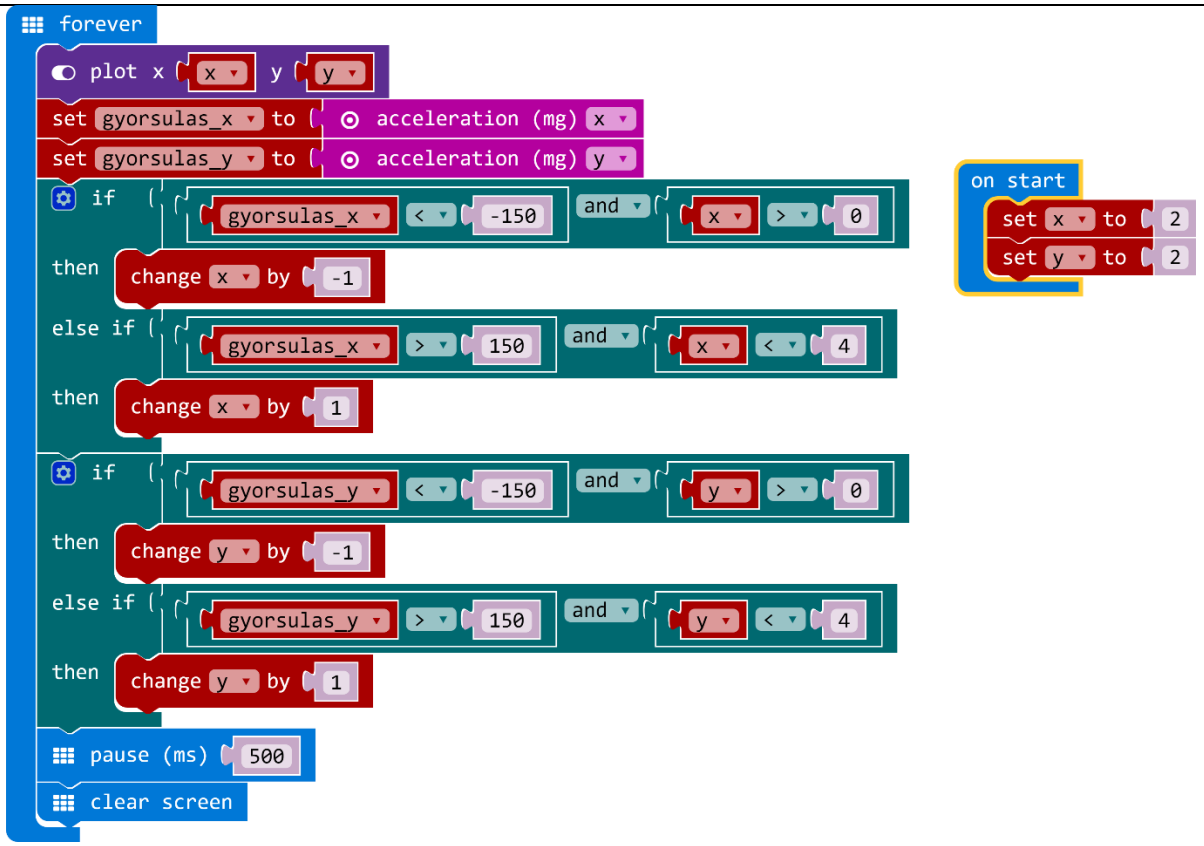
Feladat a diákok számára
Hogyan lehetne továbbfejleszteni az alkalmazást? Alakítsatok párokat és ötleteljetek! Valósítsátok meg az ötleteiteket.

Amennyiben nem sikerül a pároknak megfelelő továbbfejlesztési ötletet találnia, mi magunk is javasolhatunk témákat:

- A vigyorgó fej megjelenése után egy grafikon is jelenjen meg az ugrások számáról
- Amennyiben a micro:bitek elemtartóval is rendelkeznek, akkor jobban eltávolodhatunk a számítógépektől. Ne a felfele ugrást, hanem az előre ugrást mérjük az alkalmazással.
- Tároljuk el, hogy mi volt a legnagyobb gyorsulási érték, és ezt a „B” gomb megnyomásával írassuk ki a kijelzőre.
- A kijelzőn véletlenszerűen jelenjen meg felfele, vagy lefele mutató nyíl. Amikor felfele nyíl látható, akkor az ugrásainkkal növelhetjük a pontszámot. Lefele nyíl esetén ne ugráljunk, mert akkor csökken a pontszámunk!

4. Tojás és kanál játék

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_6u2XadUXRfkP
Sokan ismerik (és talán játszották is) 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:bitre hangolva (Az eredeti alkalmazás a https://makecode.microbit.org/examples/egg-and-spoon címen található)
Osszuk meg a diákokkal az alábbi alkalmazást és beszéljük meg, hogy mit csinál a kód: https://makecode.microbit.org/_6u2XadUXRfkP



Kezdetben a kijelző közepén gyújtunk ki egy pontot. Mérjük az x és y irányú gyorsulást, és ezek értékétől függően elmozdítjuk a pontot jobbra, balra, felfele vagy lefele.

Próbáljuk ki az alkalmazást! Amennyiben a micro:bitek elemtartóval rendelkeznek, a számítógéptől el is távolodhatunk. Készítsünk egy rövid szlalom pályát, amelyet úgy kell teljesíteni, hogy a pont a kijelzőn ne érje el a szélső sorokat, oszlopokat.

5. Saját alkalmazás fejlesztése a szenzorok felhasználásával

Feladat a diákok számára

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

Programozzunk micro:biteket!

Amennyiben az ötletelés nem vezet eredményre, mi magunk is javasolhatunk a pároknak megvalósítható alkalmazásokat.

- Készítsünk olyan szívdobbanás animációt, amelyben annál gyorsabban a ver a szív, minél többet ugráltunk a micro:bitet kézben tartva. Az animációt hang is kísérje, a hang magassága is utaljon az ugrások számára (mély hang = kevés ugrás, magas hang = sok ugrás)
- Korábban készítettünk olyan animációt, amelyben az emeletes ház ablakait gyújtottuk ki. Módosítsuk ezt az alkalmazást úgy, hogy a micro:bit által mért fényerősség alapján történjen a ledek kiyújtása. Ha világos van, akkor kevés led legyen kigyújtva, ha sötétebb, akkor több.

11. alkalom Kommunikáció a micro:bitek között

A micro:bitek rádiókapcsolaton képesek egymással kommunikálni.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Rádió kapcsolat az eszközök között	Tanári bemutató, új funkciók megismerése (rádió kapcsolat, hőmérséklet átküldése).	15 perc
2. Távíró (páros munka)	A diákok párokban elkészítik az alkalmazást a kiadott feladatnak megfelelően, majd kipróbálják az eredményt.	5 perc
3. Fényerősség megjelenítése (páros munka)	A diákok párokban elkészítik az alkalmazást a kiadott feladatnak megfelelően, majd kipróbálják az eredményt.	10 perc
4. Többfelhasználós játék készítése közösen (forró krumpli)	Tanári bemutató, új funkciók megismerése (szöveg átküldése másik eszköznek, a kapott érték alapján elágazás készítése). Az alkalmazás kipróbálása, verseny a párok között, ki érte el a legnagyobb pontszámot?	25 perc
5. A forró krumpli játék továbbfejlesztése	Az alkalmazás kipróbálása, verseny a párok között, ki érte el a legnagyobb pontszámot?	20 perc
6. Ötletelés	Ötleteljünk közösen arról, hogy milyen többfelhasználós játékokat lenne érdemes készíteni. Jegyezzük fel az ötleteket, későbbi megvalósításhoz.	15 perc

1. Rádió kapcsolat az eszközök között

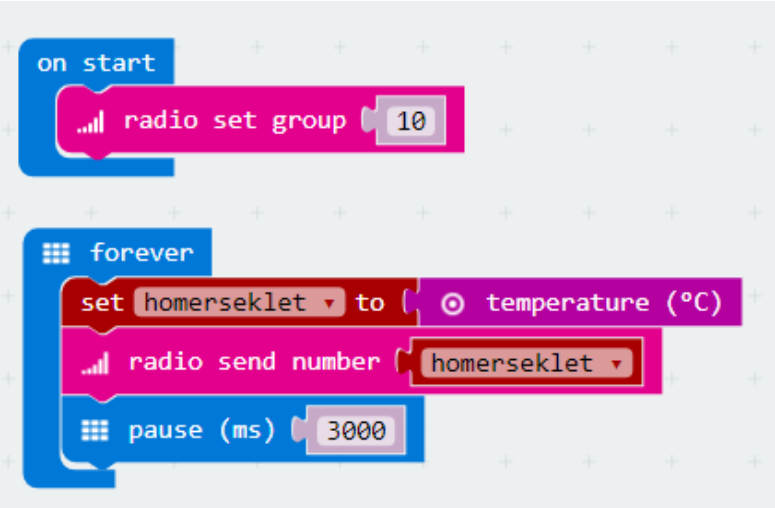
A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_TqxaqbCDsRry (Adó) https://makecode.microbit.org/_28a0CV2XF9uj (Vevő)

A micro:bitek képesek egymással kommunikálni rádiókapcsolaton keresztül. Ehhez az kell, hogy azonos csoportba szervezzük őket a **Radio** kategória **radio set group** blokkjával.

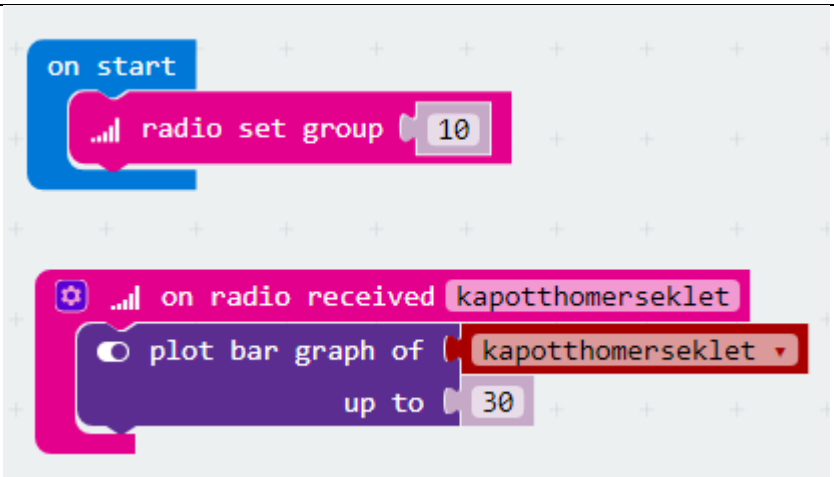
Az itt megadott szám lesz a csoport azonosítója. Ha például a szakkörön pármunkát szeretnénk végezni, és összesen 14 micro:bitünk van, akkor 7 csoportot kell alkotnunk, és a 7 csoport számára más-más számot kell beállítani. A maximális érték 255 lehet.

Készítsünk közösen egy olyan alkalmazást, amelyben az egyik micro:bit megméri a hőmérsékletet, és átküldi ezt az értéket a másik micro:bitnek, ami pedig megjeleníti azt.

Készítsük el először az adó alkalmazást. Értéket **Radio** kategória **radio send number** blokkjával tudunk átküldeni a másik eszköz számára.



Most készítsük el a vevő alkalmazást is! Itt ugyanazt a számot (10) kell azonosítóként használni, mint az előbbi esetben. A **Radio** kategóriában található **on radio received** blokk segítségével tudjuk meghatározni, hogy mi történjen akkor, ha rádió kapcsolaton kaptunk egy adott értéket. Jelen esetben rajzoljuk ki azt egy diagram segítségével.



Alakítsunk párokat, és mindenki próbálja ki a fenti alkalmazást.

Amennyiben a micro:bithez elemet tudunk kötni, máris készítettünk egy időjárás állomást. Az adót elhelyezhetjük akár az osztálytermen kívül is, belül pedig látjuk a hőmérséklet változását.

2. Távíró (páros munka)

Feladat a diákok számára
Alakítsatok párokat. Készítsetek olyan alkalmazást, amely két micro:bit közti kommunikációt valósít meg a következők szerint: • Az adó alkalmazással lehessen megadni egy tetszőleges számot 0 és 15 között, 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.

3. Fényerő megjelenítése (páros munka)

Feladat a diákok számára
Alakítsatok párokat. 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 meg, hogy az összes pont legyen kigyújtva a kijelzőn, viszont a kijelző világosságértéke legyen megváltoztatva aszerint, hogy a másik micro:bit milyen értéket adott vissza. (Emlékeztető: a fényerő 0 és 255 között vehet fel értéket. A 0 érték a teljes sötétséget, a 255 a teljes fényerőt jelenti.)

4. Többfelhasználós játék készítése (forró krumpli)

A tanár által elkészítendő/bemutatandó mintaprojekt
https://makecode.microbit.org/_cDM1wxhkfd55

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

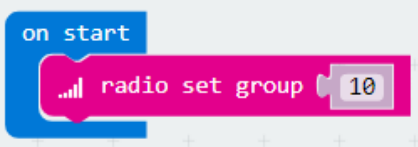
A játék leírása: Az egyik micro:bit „A” gombjának lenyomásával induljon a játék. Amikor megjelenik az ábra (forró krumpli) a kijelzőn, minél gyorsabban meg kell nyomni a „B” gombot. Ekkor az ábra a másik játékos micro:bit-jére ugrik. Neki szintén minél előbb a „B” gomb megnyomásával meg kell szabadulnia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

Ha valamelyik játékos akkor nyomja meg a „B” gombot, amikor még nincs nála a forró krumpli, az pontlevonást kap, a sikeres átküldésnél viszont növekszik a pontszáma.

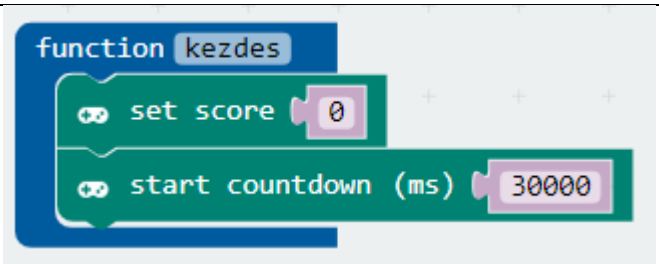
A játék 30 másodpercig tartson. A játék végén jelenjenek meg a pontszámok.

Valósítsuk meg úgy a játékot, hogy ne legyen külön adó és vevő, hanem ugyanaz a program fusson mindegyik micro:bit-en.

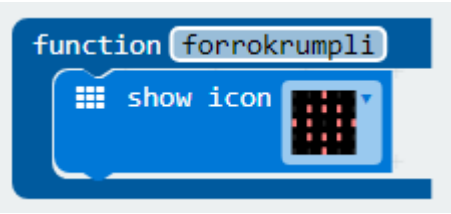
A program indításakor állítsunk be egy közös rádió csoportot a két eszköz számára.



A játék kezdetekor be kell állítanunk a pontszámot nullára, és el kell indítanunk egy visszaszámlálást 30 másodpercről. Készítsünk ehhez egy függvényt „kezdes” néven, mivel ezt a kódot többször is fel kell majd használnunk.

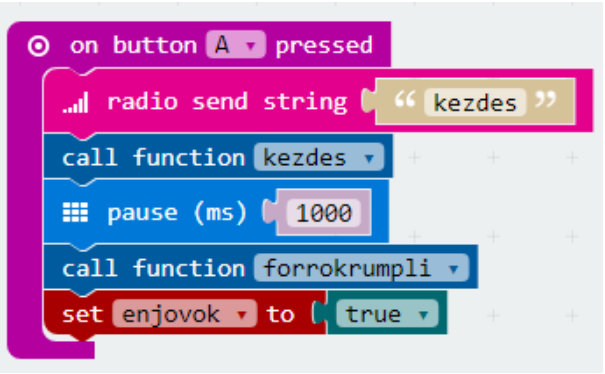


Készítsünk egy másik függvényt is „forrokrumpli” néven, amely megjeleníti a kijelzőn az ábrát.

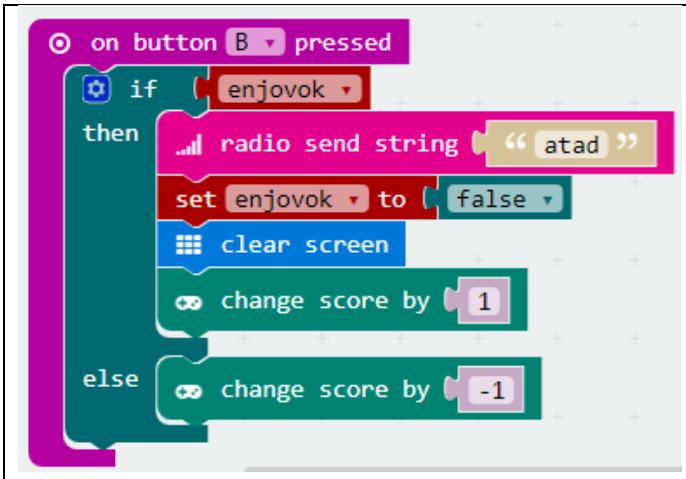


Most térjünk át arra, hogy az „A” gomb lenyomásával indítjuk el a játékot. A kezdési állapotról értesítenünk kell a másik micro:bitet is. Küldjünk át egy szöveget, mondjuk „kezdes” tartalommal. Ezek után hívjuk meg a *kezdes* függvényünket, hiszen a pontszámot le kell nullázni, és indul a visszaszámlálás. Várakozzunk picit, majd rajzoljuk ki az ábrát a képernyőre.

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



Most nézzük, hogy minek kell történnie a „B” gomb lenyomásakor. Ezzel a gombbal tudom átküldeni a másik eszközre a forró krumplit. Amikor az „enjovok” változó igaz, akkor tényleg át tudom küldeni az ábrát a másik eszközre, és megnövelhetem a pontszámot. A másik micro:bitet informálni kell arról, hogy átadom a forró krumplit, ezért át lehet küldeni neki egy „atad” üzenetet. Fontos, hogy átadás után az „enjovok” változó tartalmát állítsam át hamisra. Ha viszont akkor nyomtam meg a gombot, amikor nem én voltam soron, le kell vonni 1 pontot.



Most már csak azt kell megfogalmaznunk, hogy mi történjen az egyes üzenetek megkapásakor.

A kapott értéktől függően, vagy a kezdés függvényt kell meghívnunk, vagy meg kell jeleníteni a forró krumplit. Amikor megkaptuk a forró krumplit, szintén be kell állítanunk az „enjovok” változó értékét igazra, hiszen én vagyok a soros a továbbadásban.

Adjunk lehetőséget arra, hogy a diákok párokban kipróbálják a programot. Hívjuk fel a figyelmet arra, hogy minden párnak egyedi csoport azonosítót kell beállítani a rádiókapcsolat létrehozásakor.

5. A forró krumpli játék továbbfejlesztése

Feladat a diákok számára
Alakítsatok párokat. Ötleteljetek arról, hogy hogyan lehetne továbbfejleszteni a játékot, hogy növekedjen a játékélmény. Kezdjétek el megvalósítani az ötleteiteket.

Programozzunk micro:biteket!

Mi magunk is adhatunk ötleteket a továbbfejlesztésre, ha a pároknak nem sikerül megfelelő nehézségű továbbfejlesztési ötlettel előállni. Például:

- Kísérje hanghatás a játékot!
- Az átküldött ikont véletlenszerűen válassza ki a program, ne mindig ugyanaz az ábra jelenjen meg a kijelzőn. Amikor véletlenszerűen egy szív lett kiválasztva, akkor annak átadásakor dupla pont járjon a játékosnak.
- Ne az „A” gombbal induljon a játék, hanem az „A+B” gombbal. Kétféle ábrát küldjünk át, egy balra mutató nyilat, vagy egy jobbra mutató nyilat. A balra mutató nyilat az „A” gombbal kelljen átküldeni, a jobbra mutatót „B” gombbal. Ha nem jó gombbal küldte át a játékos, akkor csökkenjen a pontszáma.

6. Ötletelés

Feladat a diákok számára
Ötleteljünk közösen arról, hogy milyen többfelhasználós játékokat lenne érdemes készíteni. Jegyezzük fel az ötleteket, későbbi megvalósításhoz. A megvalósításra kb. 140 perc jut a következő két alkalommal, vagyis olyan nehézségű projekteket érdemes tervezni, amelyeket ennyi idő alatt le lehet fejleszteni.

12. alkalom Saját játékötletek megvalósítása (1.)

Ezen alkalommal minden diákcsoport megvalósíthatja a korábbi alkalommal megtervezett alkalmazását.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Többfelhasználós játékok fejlesztése	A diákok párokban, vagy akár nagyobb csoportokban elkészítik a tervek alapján a többfelhasználós játékokat. A tanár körbejár, segíti a munkát, válaszol a felmerülő kérdésekre, ötleteket ad a fejlesztésre vonatkozóan.	80 perc
2. Megbeszélés (meddig jutottunk, mi van még hátra)	Minden diákcsoport bemutatja, hogy hol tart a fejlesztésben, milyen funkciók készültek el, melyek vannak még hátra.	10 perc

13. alkalom Saját játékötletek megvalósítása (2.)

Ezen alkalommal be kell fejezni a játékok fejlesztését.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Többfelhasználós játékok fejlesztése	A diákcsoportok befejezik a tervek alapján a többfelhasználós játékokat. A tanár körbejár, segíti a munkát, válaszol a felmerülő kérdésekre, ötleteket ad a fejlesztésre vonatkozóan.	60 perc
2. Többfelhasználós játékok bemutatása, kipróbálása	Minden diákcsoport bemutatja az elkészült játékát. A diákok kipróbálják egymás játékait. A tanár és a diákok értékelik egymás munkáit (mi volt a jó a játékban, hogyan lehetne továbbfejleszteni, …)	30 perc

14. alkalom Barkácsoljunk együtt!

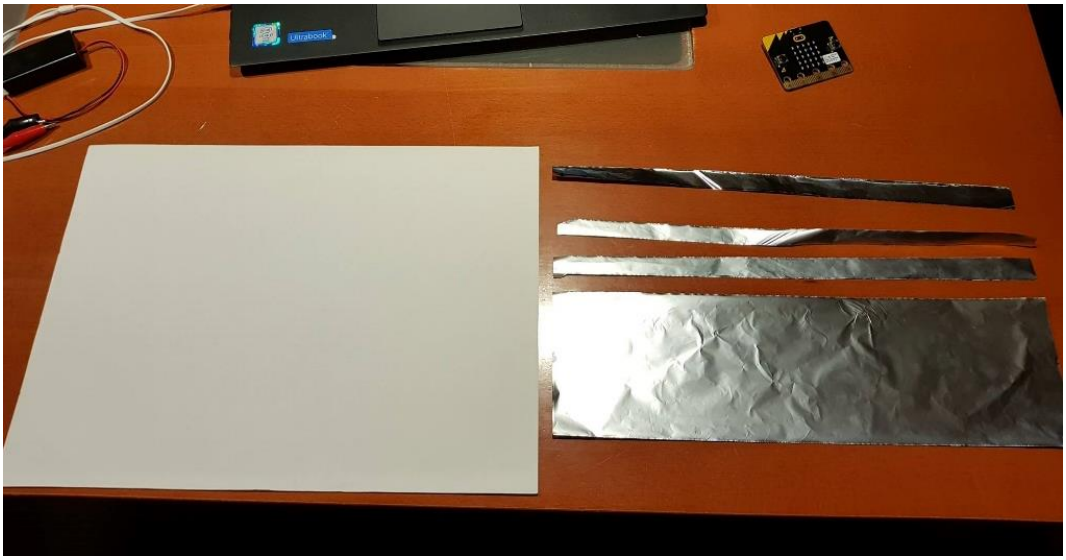
A következő néhány ötlettel megmutathatjuk a diákoknak, hogy a micro:bit kreatív felhasználásával különböző tárgyakat is készíthetünk.

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam (perc)
1. Papír zongora készítse és kipróbálása	A leírás és a tanári útmutató alapján a diák párok micro:bit által vezérelt papír zongorát készítenek és kipróbálják azt.	30 perc
2. Papír zongora (dallam lejátszás)	Alkalmazás továbbfejlesztése (dallam lejátszás) a feladat alapján, majd az eredmény kipróbálása.	10 perc
3. Papír zongora (kód módosítása)	Alkalmazás továbbfejlesztése a feladat alapján (különböző hangnemek), majd az alkalmazás kipróbálása.	10 perc
4. Csak nyugalom (türelemjáték készítése)	A leírás és a tanári útmutató alapján a diák párok egy ügyességi játékot készítenek, melynek lényege, hogy egy hurkot kell mozgatni egy hajlított drót körül úgy, hogy ne érjenek hozzá.	25 perc
5. Csak nyugalom (különböző nehézségű pályák)	Különböző nehézségű pályák készítése és azok kipróbálása.	10 perc
6. A szakkör lezárása	A tanár összefoglalja a szakkörön elért eredményeket, értékeli a diákok tevékenységeit. Amennyiben a gyerekek lelkesedése kitart és vannak még ötleteik, valamint hely, idő, energia azok megvalósítására, akkor újabb szakköri időpontokat is ki lehet tűzni, ahol mindenki megvalósíthatja kreatív ötleteit, a tanár pedig facilitátori szerepet tölt be.	5 perc

1. Papír zongora készítése

A papír zongora ötletét Fülöp Tamás tette közzé a Micro:bit műhely nevű facebook csoportban (http://bit.ly/2EhHNGv). Ezen csoportot ajánljuk minden olyan diák és kolléga figyelmébe, aki szeretne ötleteket kapni a micro:bitek felhasználására vonatkozóan, és arra buzdítunk mindenkit, hogy ötleteit ossza meg másokkal is.

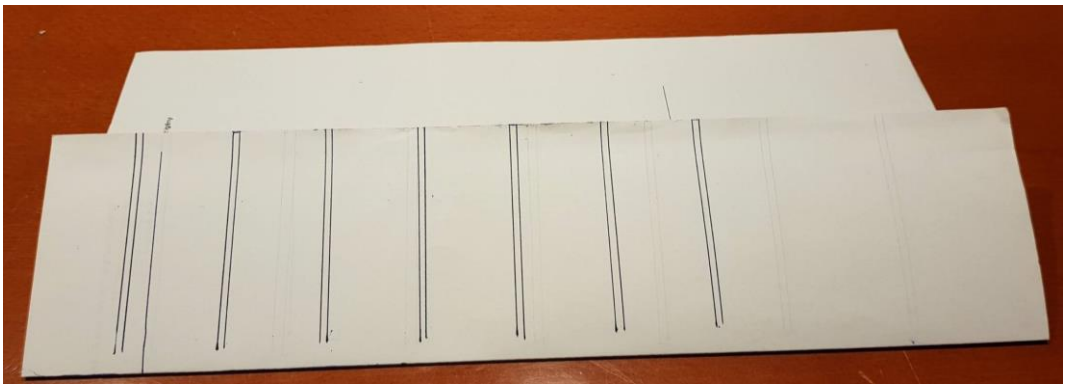
A tanár által elkészítendő/bemutatandó mintaprojekt
A következőkben a papír zongora elkészítésének lépéseit találjuk. A fotókat és a leírást Fülöp Tamás (ARM Hungary) készítette.
Hozzávalók (1 zongorához)



- 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

Az ideális az lenne, ha a tanári útmutatóval párhuzamosan a projektet párokban elkészíthetnék a diákok, vagyis ennek megfelelő számú hozzávaló szükséges.

1. lépés



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

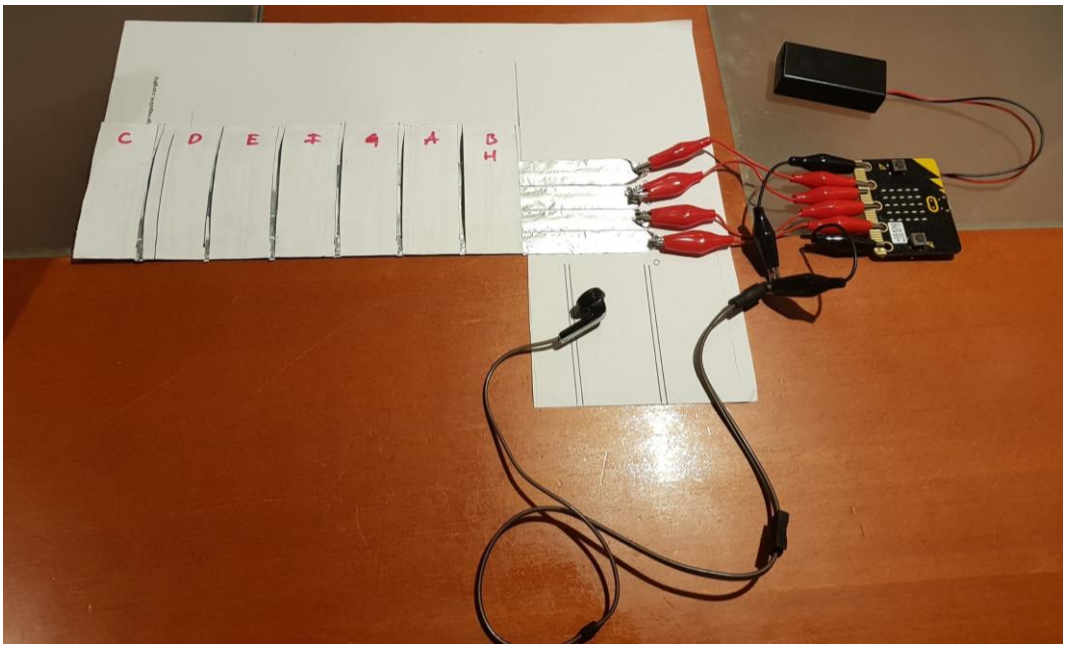
2. lépés

Ragasszuk fel a csíkokat. A vastagot ragasszuk a szélére. Ez lesz a billentyűk alján. A vastag alufóliának túl kell érnie a hajtáson. A csíkok azon végét, amelyek a microbithez fognak csatlakozni ne ragasszuk le végig.

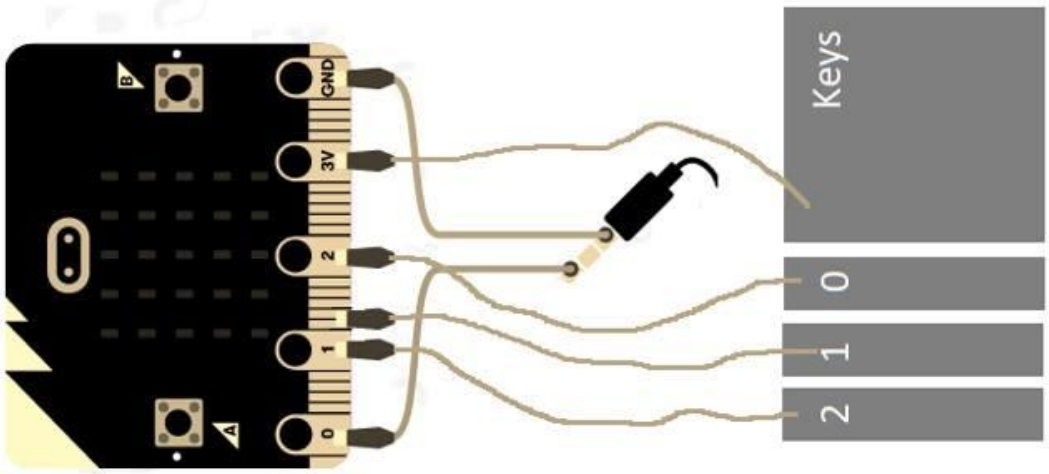
3. lépés

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

4. lépés



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

5. lépés
 Csatlakoztassuk a microbitet a zongorához és egy fülhallgatóhoz. A harmadik csipesz a pin8-ra csatlakozik.
6. lépés
Az alábbi kódot töltsük rá a micro:bitre, amellyel a zongora életre kelthető: https://makecode.microbit.org/_17mWagHDFLhr

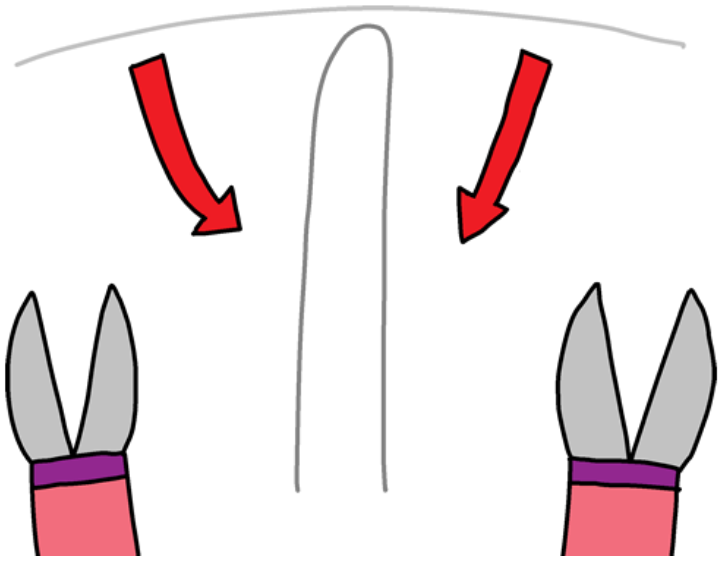
2. Zongora továbbfejlesztése (dallam lejátszás)

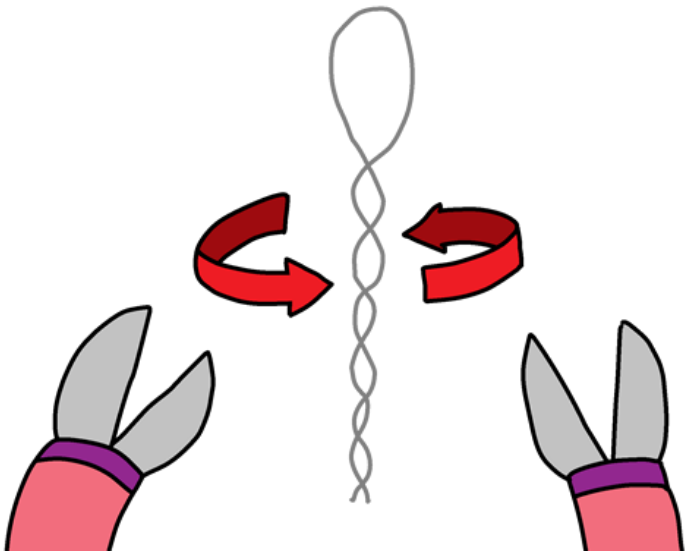
Feladat a diákok számára
Amikor megnyomjuk az „A” gombot, akkor véletlenszerűen választott dallamot játsszon le a micro:bit. Utána próbáljuk ugyanazt a dallamot elzongorázni!

3. Zongora továbbfejlesztése (hangnem módosítás)

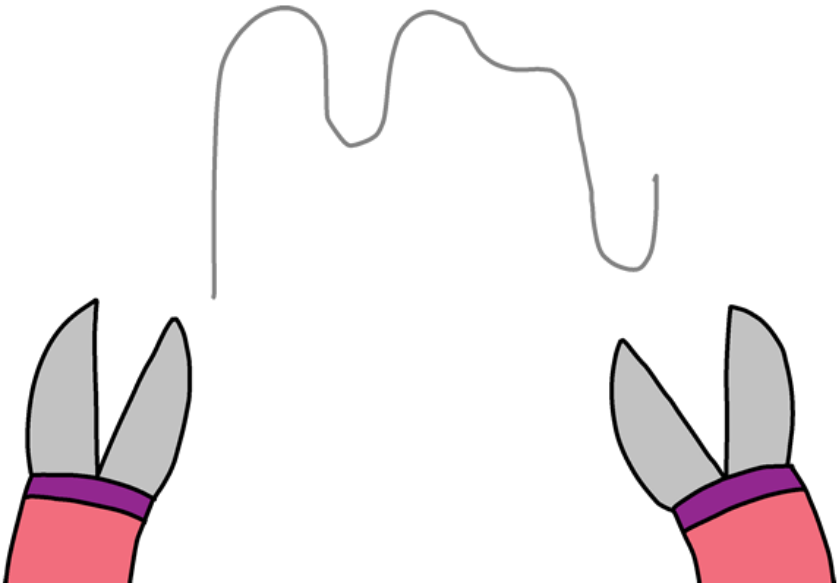
Feladat a diákok számára
Álljon össze három diákcsoport. Módosítsátok a zongorák hangnemét! Az egyik zongora alacsony, a másik közép, a harmadik magas hangon szólaljon meg. Próbáljátok mindhárom zongorán egyszerre ugyanazt a dallamot eljátszani.

4. Csak nyugalom

A tanár által elkészítendő/bemutatandó mintaprojekt
A következő ötlet a https://codeclubprojects.org/en-GB/microbit/frustration/ oldalról származik. A megvalósítandó 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 (1 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 Az ideális az lenne, ha a tanári útmutatóval párhuzamosan a projektet párokban elkészíthetnék a diákok, vagyis ennek megfelelő számú hozzávaló szükséges.
1. lépés (Hurok elkészítése)
 Készítsük el a hurkot. Vegyünk egy kb. 20 cm hosszú drót darabot, majd hajlítsuk meg gy, hogy a két szár kb. egyenlő hosszúságú legyen. Ezután csavarjuk össze a két szárát.

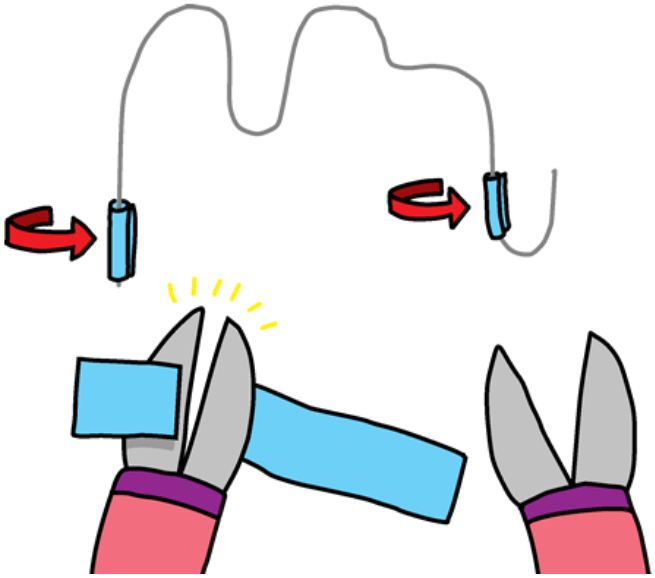


2. lépés (A pálya elkészítése)



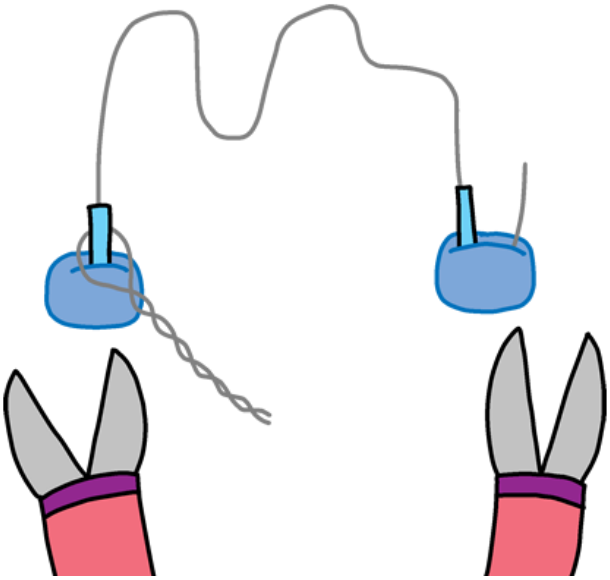
A (maradék) 30 cm hosszú drótból hajtogassunk egy pályát, e körül kell majd a hurkot végig vezetnünk. A drót egyik végét hajlítsuk felfelé.

3. lépés (szigetelőszalag felhelyezése)



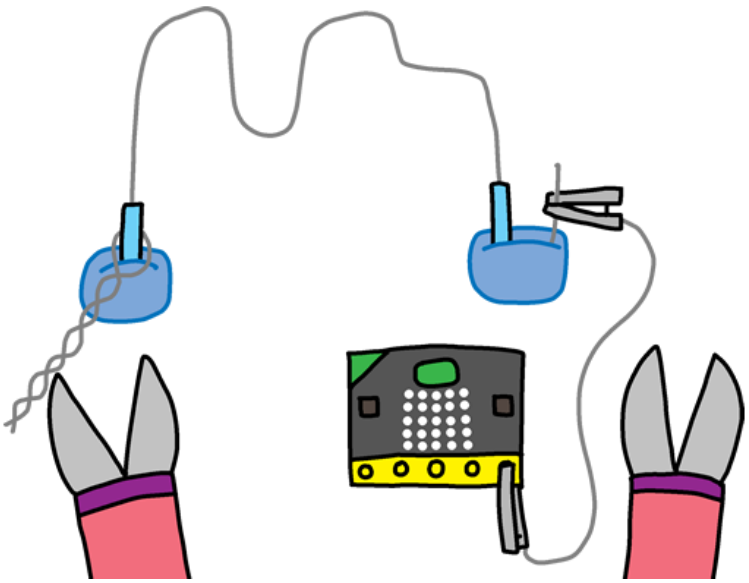
A szigetelőszalagot a drót mindkét végénél ragasszuk fel úgy, hogy maradjon csupasz drótvég is.

4. lépés

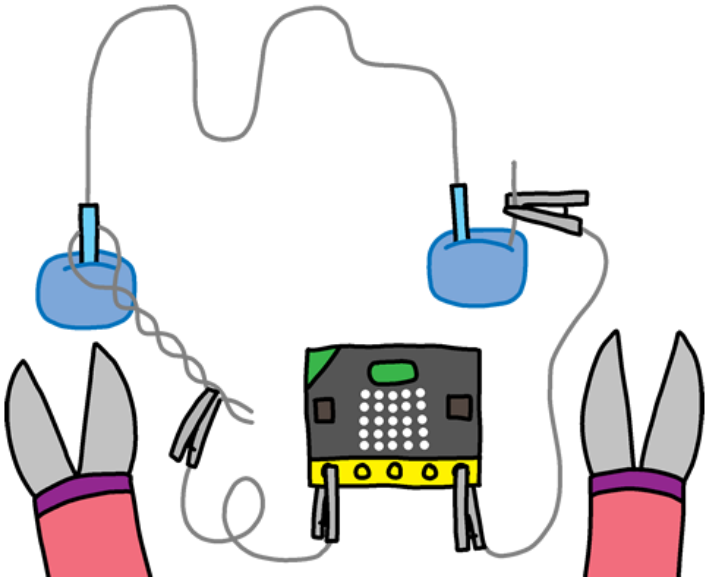


A hurkot fűzzük át a dróton, és a drót két végét helyezzük el a gyurmában (vagy hungarocellben)

5. lépés (micro:bit csatlakoztatása)



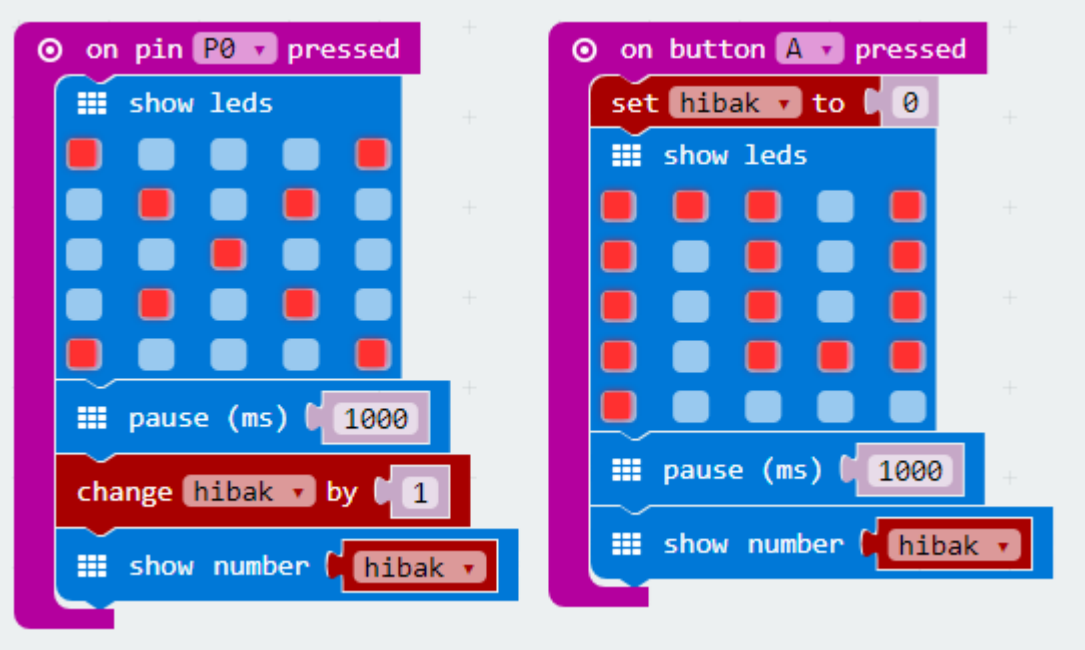
A micro:bit GND (föld) kivezetését csatlakoztassuk a krokodilcsipesszel a drót jobb oldalához.



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

6. lépés

Készítsük el a következő kódot

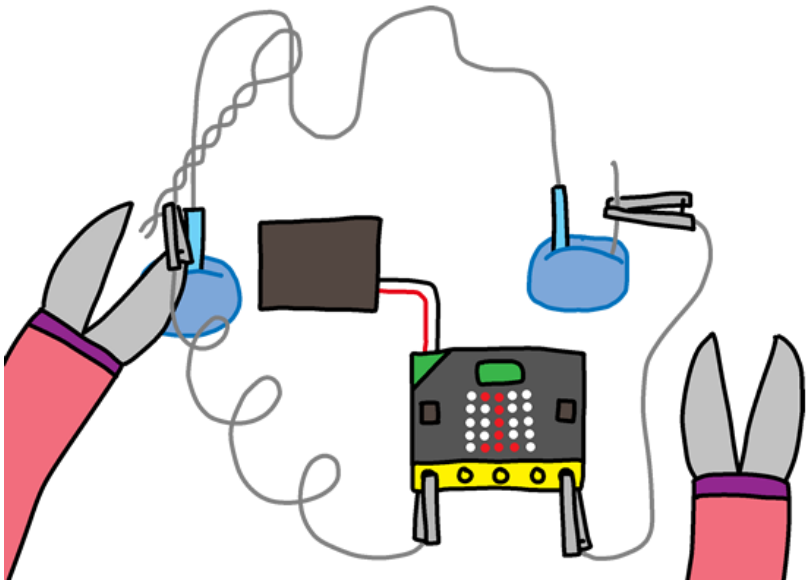


Amikor a hurokkal hozzáérünk a vezetékhez, akkor jelenítsünk meg egy X jelet a kijelzőn, várjunk picit, és növeljük meg a hibák számát eggyel, majd jelezzük ki ezt a számot.

Az „A” gomb megnyomásakor induljon a játék. A hibák számát nullázzuk le, jelenítsünk meg egy ábrát, amely a pályát jelképezi, várjunk egy picit, és mutassuk meg a hibák számát.

A kész alkalmazás elérhetősége: https://makecode.microbit.org/_8PsKRXMHfbgv

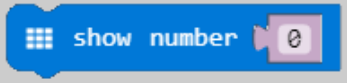
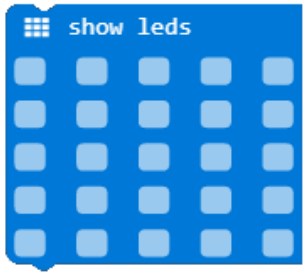
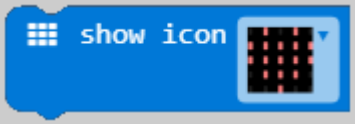
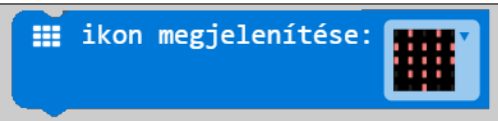
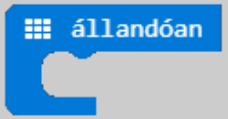
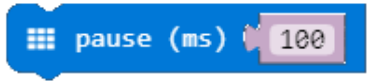
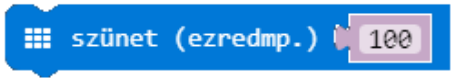
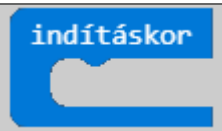
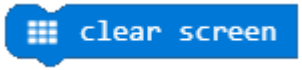
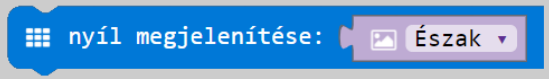
Nem marad más hátra, mint próbáljuk ki a játékot.

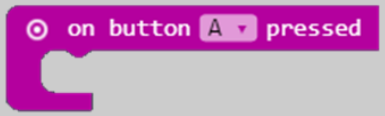
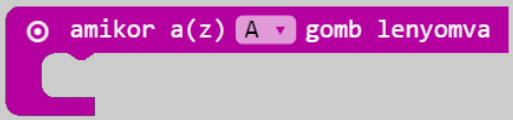
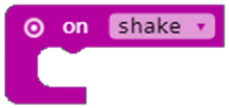
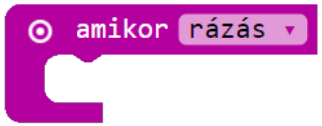
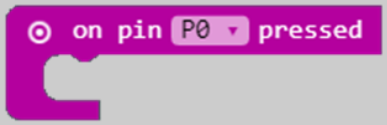
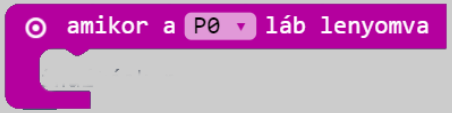
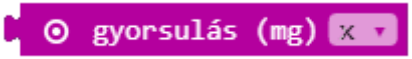
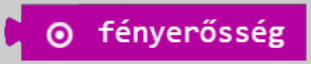
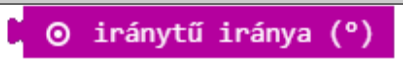
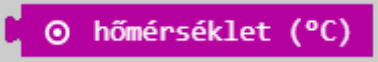


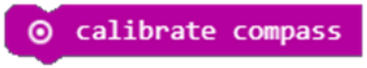
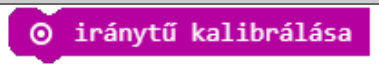
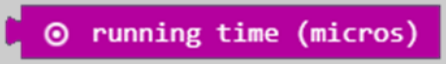
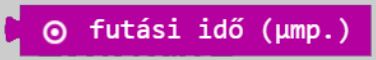
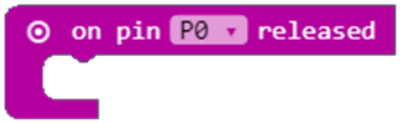
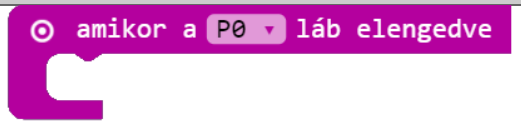
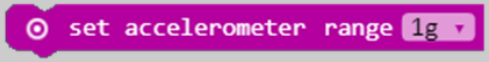
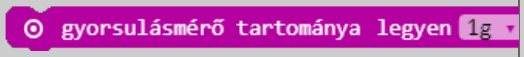
5. Csak nyugalom (különböző nehézségű pályák)

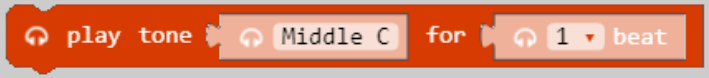
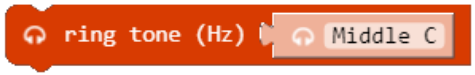
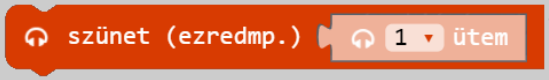
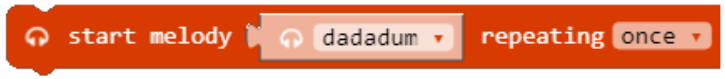
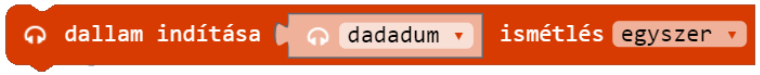
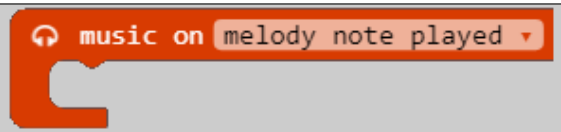
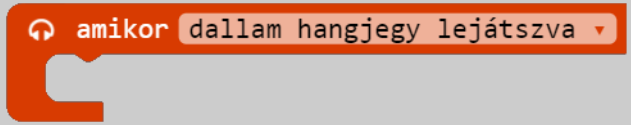
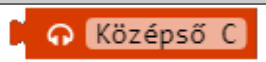
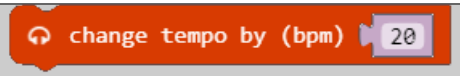
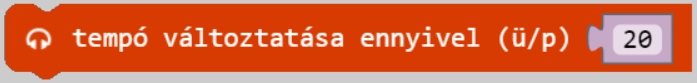
Feladat a diákok számára
Készítsetek különböző nehézségű pályákat, majd próbáljátok ki az összes pályát!

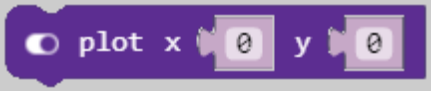
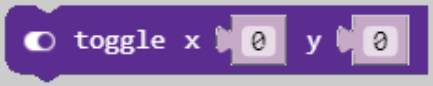
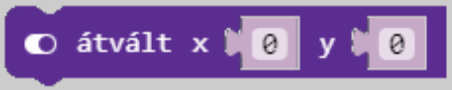
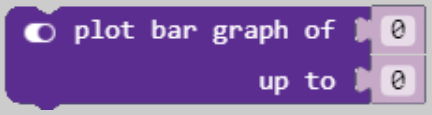
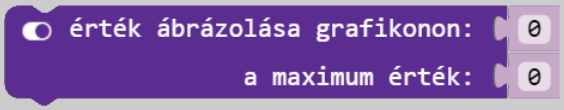
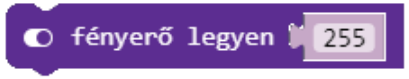
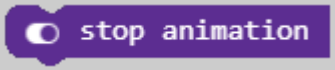
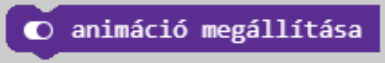
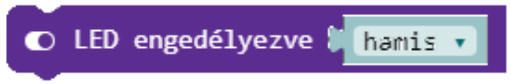
MELLÉKLETEK

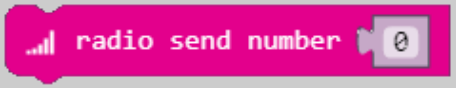
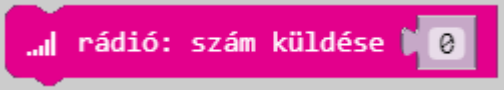
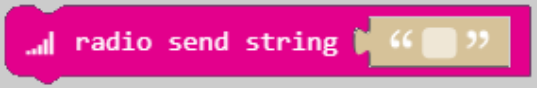
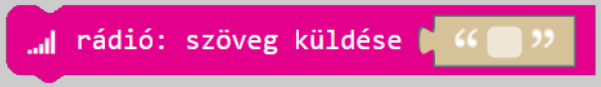
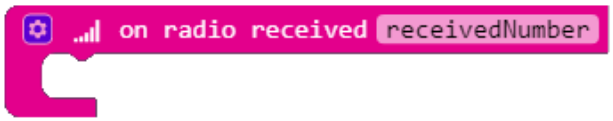
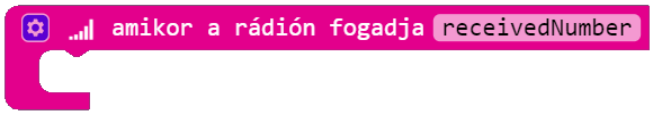
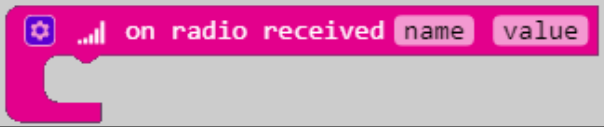
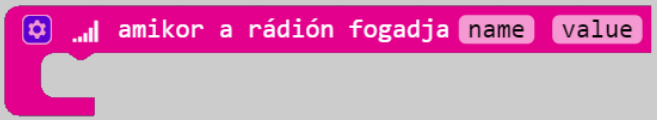
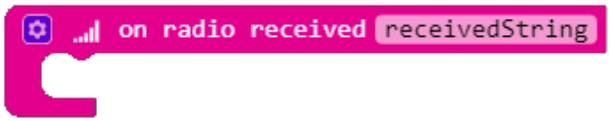
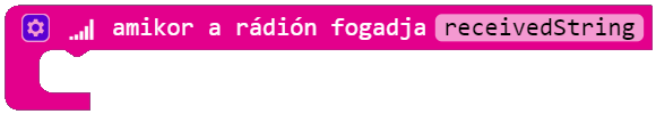
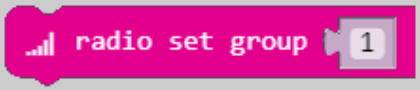
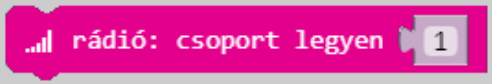
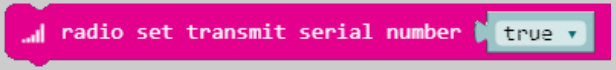
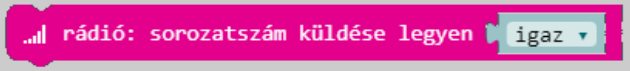
Basic kategória	Alapok kategória	Leírás
		**Szám** kiírása a kijelzőre
		A kiválasztott **Ledek bekapcsolása**
		A **kiválasztott ikon** megjelenítése
		**Szöveg kiírása** a kijelzőre
		Vezérlő blokk a **háttérben** („állandóan”) futó kód számára
		**Szünet** a megadott ideig (ezredmásodpercben)
		A blokk tartalma a program **indításakor** fut le
		A kijelző **letörlése**
		Különböző égtájak felé mutató **nyilakat** jelenít meg a kijelzőn (Sorrendben: Észak,Észak-kelet, Kelet, Dél-kelet, Dél, Dél-nyugat, Nyugat, Észak-nyugat)

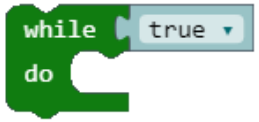
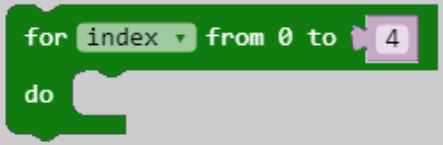
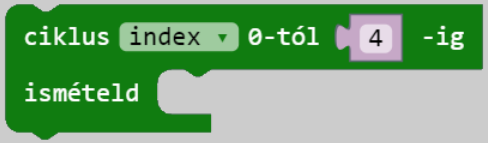
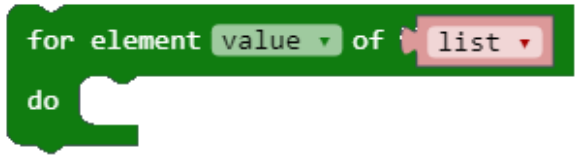
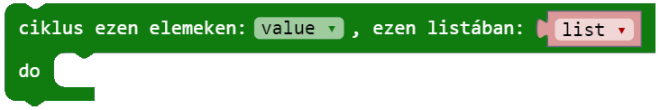
Input kategória	Bemenet kategória	Leírás
		Az A, B, A+B gombok **lenyomásakor** hajtódik végre
		A rázás, logó fent, logó lent, képernyő fent, képernyő lent, balra döntés, jobbra döntés, szabadesés, 3g, 6g, 8g **esemény bekövetkezésekor** hajtódik végre
		A különböző **lábak** (P0,P1,P2) **megnyomásakor (érintésekor)** hajtódik végre
		Ha az A, B, A+B gomb **le van nyomva, igaz** értéket ad vissza
		Ha az P0, P1, P2 láb **le van nyomva (meg van érintve), igaz** értéket ad vissza
		A **gyorsulás** értékét adja vissza x,y,z tengely szerint, vagy az ezekből számított erősséget (mg).
		A **fényerősség szintjét** adja vissza (0 sötét – 255 teljes megvilágítás)
		Az **iránytű** irányát adja vissza fokban (0 és 359 között)
		A mért **hőmérsékletet** adja vissza Celsius fokban.
		Az eszköz **dőlésének** mértékét adja vissza fokokban. (pitch=fel/le bólintás, roll=balra/jobbra döntés)
		A **mágneses térerő** mértékét adja vissza x,y,z tengely szerint, vagy az ezekből számított erősséget.

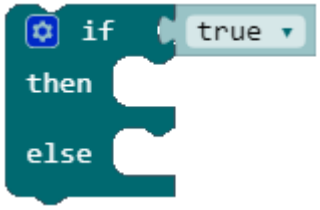
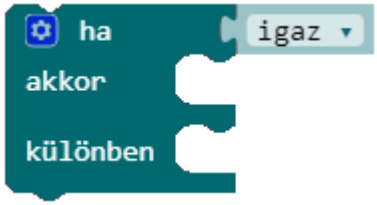
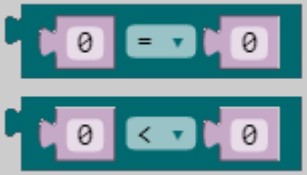
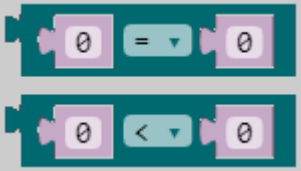
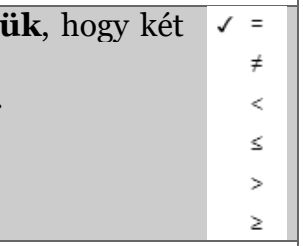
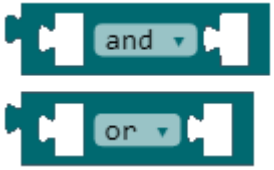
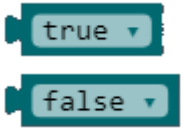
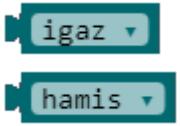
Input kategória (folytatás)	Bemenet kategória (folytatás)	Leírás
		A program elindítása óta **eltelt időt** adja vissza (**ms**)
		Az **iránytű kalibrálási** folyamatának elindítása
		A program elindítása óta **eltelt időt** adja vissza (**micros**)
		A különböző **lábak** (P0,P1,P2) **elengedésekor** hajtódik végre
		A **gyorsulásmérő mérési tartományát** lehet beállítani (1g, 2g, 4g, 8g)

Music kategória	Zene kategória	Leírás
		Az adott **hangot** a megadott **ütemmel** lejátssza
		Az adott **hangot** megszólaltatja
		**Szünetet** (csend) tart a megadott ütemig
		A kiválasztott **dallamot** lejátssza a megadott **ismétléssel** (egyszer, állandóan, háttérben egyszer, háttérben állandóan)
		A **zene állapotára** vonatkozó vezérlőblokk (pl. dallam elindítva, dallam véget ér, dallam ismételve stb.)
		Visszaadja az adott hang **frekvenciáját**
		Visszaadja a megadott **ütem hosszát** ezredmásodpercben
		Visszaadja a **tempó értékét** (ütem / perc)
		**Növeli** (pozitív szám) / **csökkenti** (negatív szám) a **tempót** az adott értékkel (ütem / perc)
		**Beállítja a tempót** az adott értékre (ütem / perc)

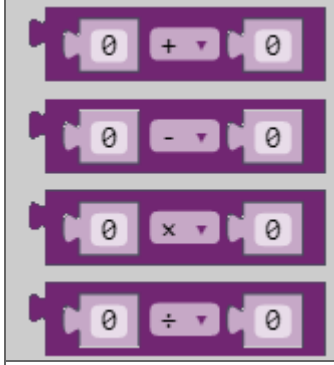
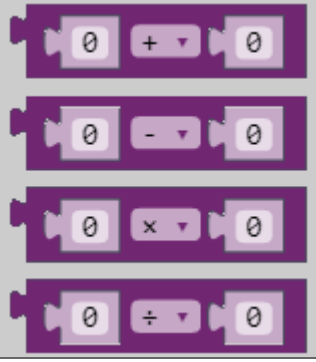
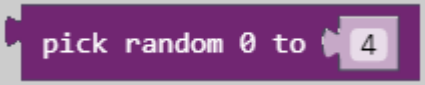
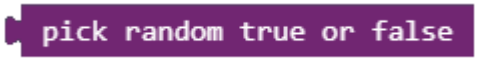
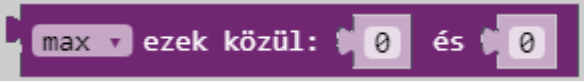
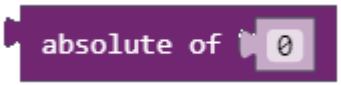
Led kategória	Led kategória	Leírás
		Adott koordinátájú pont **felkapcsolása**
		Adott koordinátájú pont **lekapcsolása**
		Adott koordinátájú pont **állapotának átváltása**
		Adott koordinátájú pont **állapotának lekérdezése** (igaz, ha be van kapcsolva)
		**Grafikon rajzolása**. Az első paraméter az ábrázolandó érték, a második pedig a maximális érték.
		Adott koordinátájú **pont felkapcsolása** a megadott **fényerősséggel**
		A **fényerősség** lekérdezése
		A **fényerősség beállítása** (0 – 255 között)
		Az **animáció megállítása**
		A LED kijelző **engedélyezése**

Radio kategória	Rádió kategória	Leírás
		**Szám küldése**
		**Név és érték** küldése
		**Szöveg** küldése
		Vezérlőblokk (**szám fogadásakor** hajtódik végre)
		Vezérlőblokk (**változó és érték fogadásakor** hajtódik végre)
		Vezérlőblokk (**szöveg fogadásakor** hajtódik végre)
		A rádiókapcsolat **csoportazonosítójának** beállítása
		Az **adási teljesítmény** beállítása (7 a maximum)
		Az eszköz **sorozatszámának** elküldése minden adatcsomagban
		A kapott csomag **soros vonalra** írása

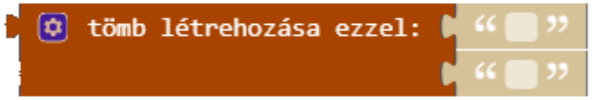
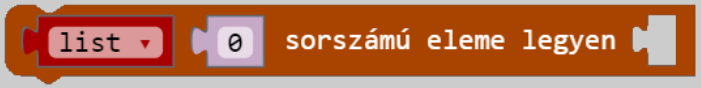
Loops kategória	Ciklusok kategória	Leírása
		A blokk tartalmát pontosan a **megadott számú alkalommal** ismétli meg
		Elöltesztelő ciklus. Addig ismétli a blokk tartalmát, **amíg a megadott feltétel igaz**
		**Számlálós ciklus**. A ciklusváltozó **0-tól** a **megadott számig** veszi fel az értékeket. Mivel nullától indul a számolás, az ismétlés eggyel többször történik, mint a megadott szám.
		Bejáró ciklus. A ciklus **a megadott lista elemein** megy végig

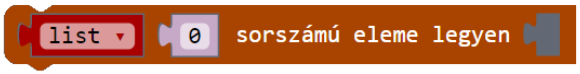
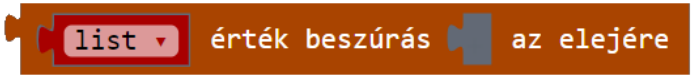
Logic kategória	Feltételek kategória	Leírás
		**Elágazás** (Ha *igaz a feltétel*, akkor ...)
		**Elágazás különben ággal** (Ha *igaz a feltétel*, akkor ... **különben** ....)
  		Ez a két blokk ugyanarra a célra szolgál. **Eldönthetjük**, hogy két szám egyenlő-e, az első kisebb-e, mint a második, stb. A kiválasztható **relációjelek** jobb oldalon látszódnak.
		Ezzel a két blokkal l**ogikai kifejezést** adhatunk meg, ami lehet **és** (*and*), illetve **vagy** (*or*) kapcsolat.
		**Tagadás** logikai művelet. Igazból hamist, hamisból igazat állít elő-
		**Igaz** (true), illetve **hamis** (false) értéket állíthatunk be

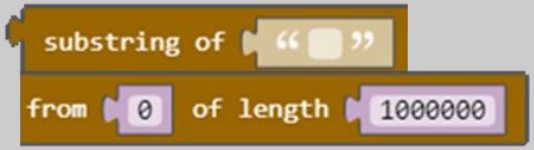
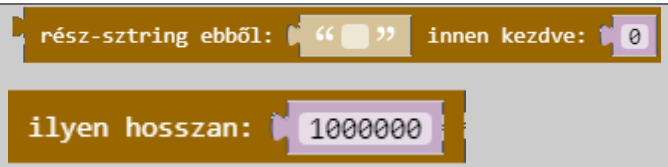
Variables kategória	Változók kategória	Leírás
		**Változó létrehozása**
		**Változó**
		**Változó értékének beállítása**
		Számot tartalmazó **változó értékének növelése / csökkentése** (attól függően, hogy a megadott szám pozitív vagy negatív)

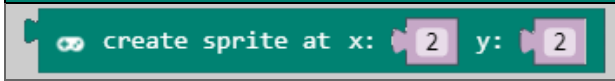
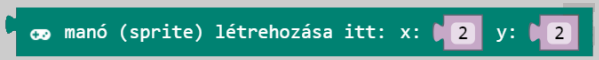
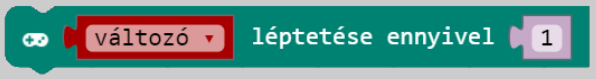
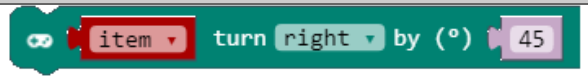
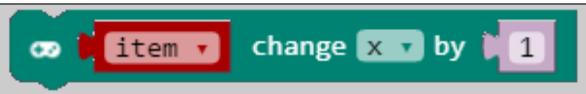
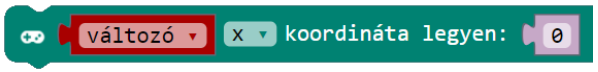
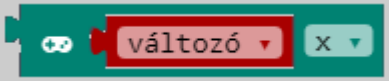
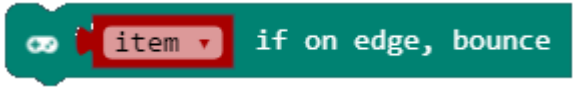
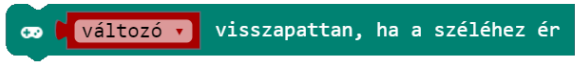
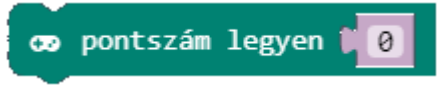
Math kategória	Matematika kategória	Leírás
		**Alapműveletek** elvégzésére szolgáló blokkok (összeadás, kivonás, szorzás, osztás). A legördülő menüből a hatványozás (**) is kiválasztható.
		Egy megadott **egész szám**
		**Véletlenszám** létrehozása 0 és a megadott szám között
		**Véletlen** választás **igaz** és **hamis** közül
		Osztási művelet **maradékát** adja vissza
		Két szám közül a **kisebbet** (minimum) adja vissza
		Két szám közül a **nagyobbat** (maximum) adja vissza
		A megadott szám **abszolút értékét** adja vissza
		Egy **karaktert** ad vissza az **ASCII** kódtábla alapján

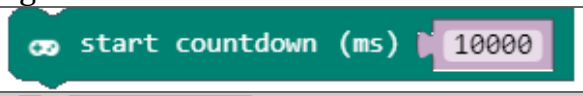
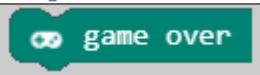
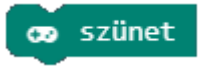
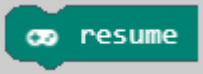
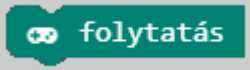
Functions kategória	Függvények kategória	Leírása
		**Függvény létrehozása**
		A létrehozott **függvény meghívására** szolgáló blokk.

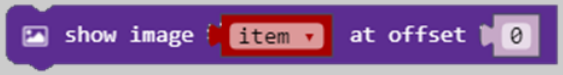
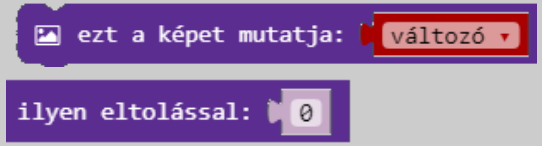
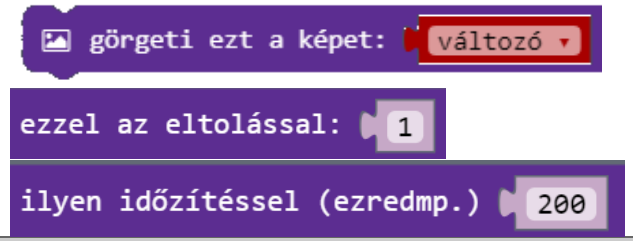
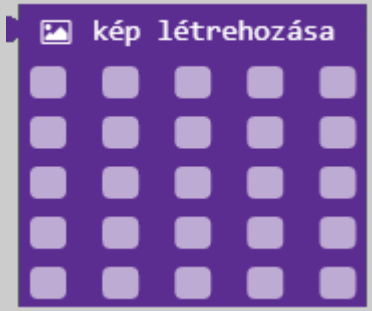
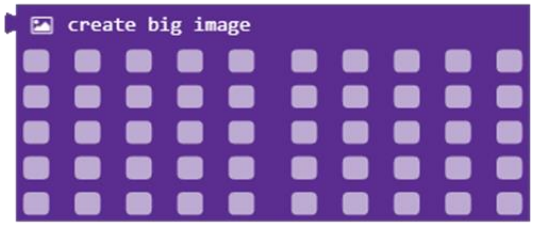
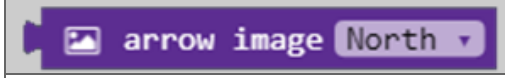
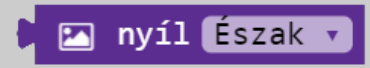
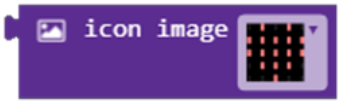
Arrays kategória	Tömbök kategória	Leírása
		**Tömb létrehozása** a megadott **szám** értékekkel
		**Tömb létrehozása** a megadott **szöveges** értékekkel
		Visszaadja a **tömb hosszát**
		A lista **adott sorszámú elemét** adja vissza
		A lista **adott sorszámú elemének beállítja** a második paraméterben megadott értéket
		A lista **végére beszúrja** a megadott értéket

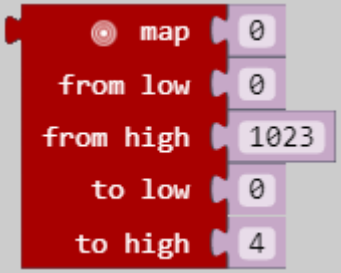
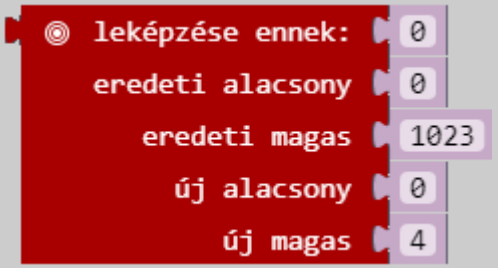
		Visszaadja **a lista utolsó elemét**, és eltávolítja azt a listából
		A megadott pozícióra **beszúr a listába** egy adott értéket
		Visszaadja a **lista első elemét**, és eltávolítja azt a listábó
		A **lista elejére beszúrja** a megadott értéket
		A megadott **értéket keresi** a listában, és visszaadja a sorszámát
		**Törli a listából** a megadott sorszámú elemet
		**Megfordítja** a lista tartalmát

Text kategória	Szöveg kategória	Leírása
		Egy **szöveget** adhatunk meg
		Visszaadja a **szöveg hosszát**
		**Összefűzi** a megadott szövegeket
		A megadott szövegből **visszaadja** az **adott sorszámú** karaktert (0-tól sorszámozódik)
		A szöveg egy **megadott hosszúságú** (*length*) részét adja vissza az adott sorszámú elemtől
		A szöveget **számmá alakítja**
		**Összehasonlítja** két szöveget. A visszaadott érték -1,0,1 lehet, attól függően, hogy a szöveg kisebb, egyenlő, vagy nagyobb a másik szöveghez képest. Az összehasonlítás az ábécésorrend, illetve a hossz alapján történik.

Game kategória	Játék kategória	Leírása
		Létrehoz egy spriteot (manót) a megadott koordinátán
		Törli az adott nevű spriteot (manót)
		Az adott nevű elemet 1-el előre lépteti
		Az adott nevű elemet elfordítja a megadott szöggel jobbra/balra *(right/left)*
		Az adott nevű elem x/y koordinátáját megnöveli (negatív érték esetén csökkenti)
		Az adott nevű elem x/y koordinátáját beállítja
		Visszaadja az adott nevű elem x/y koordinátáját
		Visszaadja, hogy az adott elem hozzáért-e egy másik elemhez (igaz, ha igen)
		Visszaadja, hogy az adott elem hozzáért-e az oldalfalhoz
		Beállítja, hogy az adott nevű elem pattanjon vissza, ha a falhoz ér
		A pontszámot növeli/csökkenti (attól függően, hogy pozitív, vagy negatív az adott szám)
		A pontszámot beállítja a megadott értékre
		Visszaadja a pontszámot

		Elindít egy visszaszámlálást a megadott időtartamra (ezredmásodpercben).
		A játék befejezése
		A játék szüneteltetése
		A játék folytatása

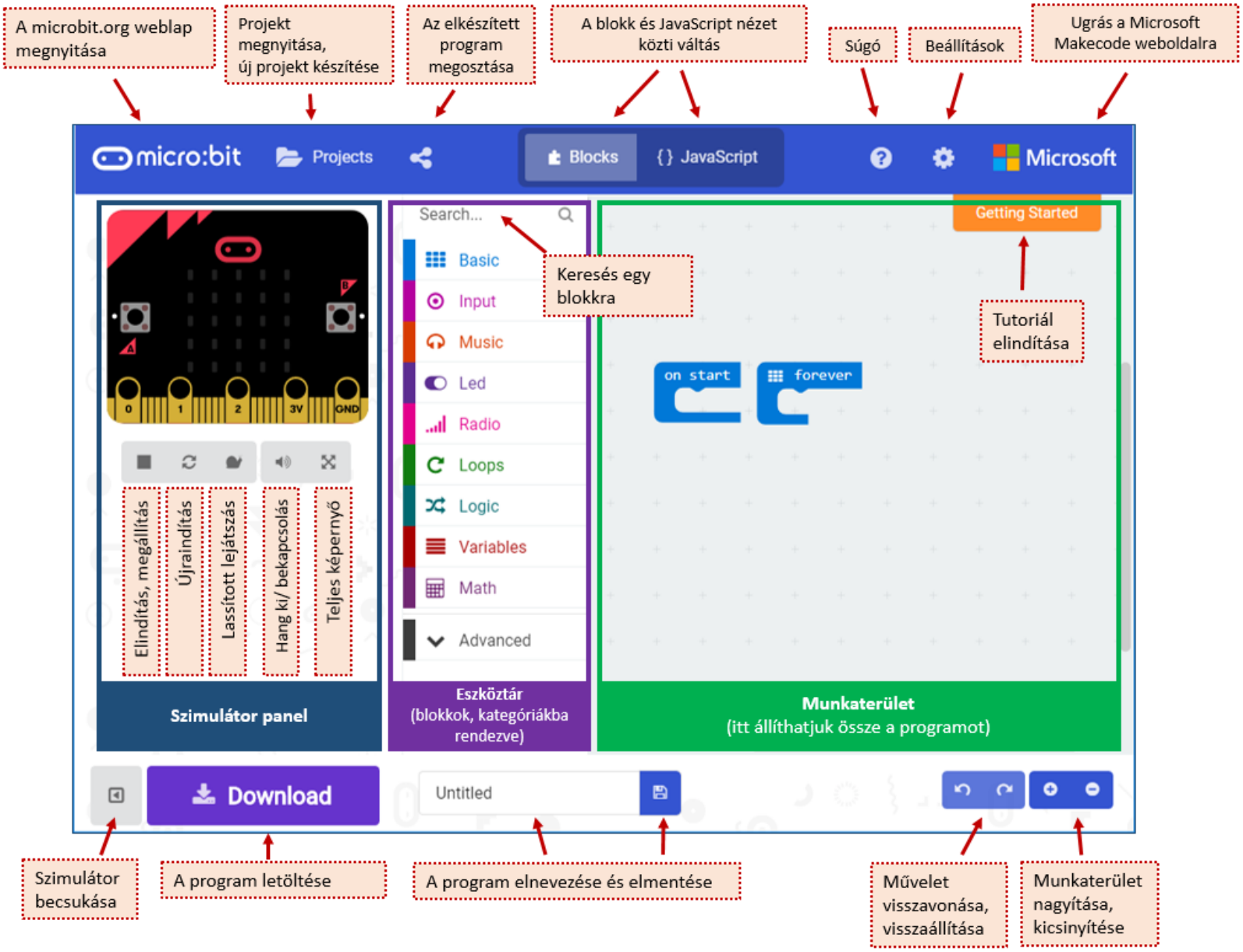
Images kategória	Képek kategória	Leírása
		**Megjeleníti** az adott képet, a megadott eltolással
		**Gördíti** az adott képet a megadott lépéssel és gyorsasággal
		**Létrehoz** egy **képet**
		**Létrehoz** egy **nagy** képet
		**Nyíl** megjelenítése az égtájak szerint
		**Ikon** megjelenítése
		Visszaadja az adott **égtájhoz** tartozó **sorszámot**

Pins kategória	Csatlakozó lábak kategória	Leírása
		A megadott csatlakozóról digitálisan olvas
		Digitálisan ír a megadott csatlakozóra
		A megadott csatlakozóról analóg módon olvas
		Analóg módon ír a megadott csatlakozóra
		Az első értéket átskálázza a megadott intervallumról egy másik intervallumra
Ebben a kategóriában, további, haladó blokkok is elérhetőek, de ezekre ezen anyag keretében nem térünk ki.		

Egyéb blokkok

Az alapértelmezetten megjelenő blokkok mellett **speciális csomagokat** is használhatunk (pl. motorok vezérléséhez). Ehhez az Add Package (Csomag hozzáadása) gombra kell kattintani.

A makecode programozási felület részei (angol felület)



A makecode programozási felület részei (magyar felület)