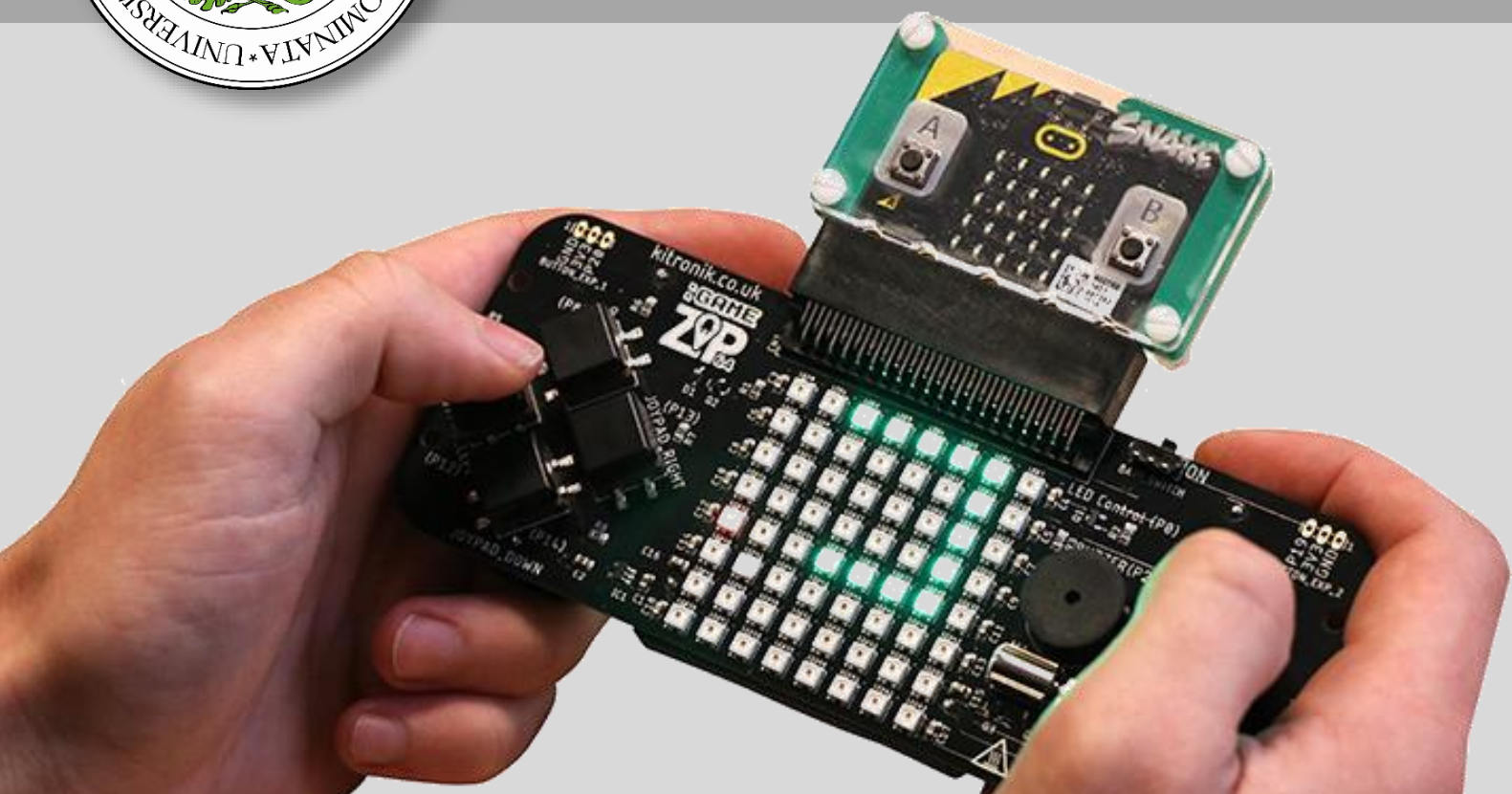




Belépő a tudás közösségébe

Szakköri segédanyag tanárok számára



Programozzuk micro:biteket a :GAME ZIP 64 játékplatform felhasználásával

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 2020-ban.



Eötvös Loránd Tudományegyetem
Informatikai Kar



MAGYARORSZÁG
KORMÁNYA

SZÉCHENYI 2020

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

Programozzuk micro:biteket a :GAME ZIP 64 játékplatform felhasználásával!

Szerző

Dr. Abonyi-Tóth Andor

Felelős kiadó

ELTE Informatikai Kar

1117 Budapest, Pázmány Péter sétány 1/C.

ISBN szám

ISBN 978-963-489-196-3

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

Tartalomjegyzék:

| | |
|----------------------------------------------------------------------------|-----------|
| A :GAME ZIP 64 készlet bemutatása | 4 |
| Az eszköz felépítése | 4 |
| Programozás a makecode felületen | 5 |
| Szükséges blokkok hozzáadása | 5 |
| A :GAME ZIP64 kategóriában elérhető blokkok bemutatása..... | 5 |
| Egyszerű demó projektek a korábban ismertetett blokkok megismeréséhez..... | 8 |
| 1. Színek váltakozása..... | 8 |
| 2. Szivárvány színeinek megjelenítése | 8 |
| 3. LED-ek színeinek beállítása koordinátánként..... | 9 |
| 4. Véletlenszerűen kiválasztott LED-ek, véletlenszerű színnel | 9 |
| 5. Pontok eltolása | 10 |
| 6. Rezgőmotor és zümmer használata..... | 10 |
| A szakkör felépítése..... | 11 |
| A témák..... | 12 |
| Felépítés, jelölések | 12 |
| 1. alkalom (Ismerkedés az eszközzel)..... | 13 |
| Munkavédelmi előírások, az eszköz bemutatása | 13 |
| Munkavédelmi előírások..... | 13 |
| Az eszköz bemutatása | 15 |
| A :GAME ZIP 64 lehetőségei a gyakorlatban | 16 |
| Közlekedési lámpa szimuláció készítése | 18 |
| Közlekedési lámpa szimuláció egyéni továbbfejlesztése | 19 |
| Közlekedési lámpa szimuláció páros továbbfejlesztése (haladóknak) | 19 |
| 2. alkalom (Játék a színekkel) | 20 |
| Játék a színekkel..... | 21 |
| Játék a színekkel - továbbfejlesztés | 24 |
| Játék a színekkel – saját játék fejlesztése pármunkában | 24 |
| 3. alkalom (Irány az úr!) | 25 |
| Úrhajós játék..... | 26 |
| Úrhajós játék - továbbfejlesztés | 29 |
| 4. alkalom (Kapj el!)..... | 30 |
| Kapj el!..... | 31 |
| Alap játék továbbfejlesztése..... | 33 |
| 5. alkalom (Kígyó)..... | 34 |
| Kígyó - alapjáték..... | 35 |

| | |
|--------------------------------------------------------|-----------|
| Kígyó alap játék továbbfejlesztése | 38 |
| 6. alkalom (Torpedó) | 39 |
| Torpedó - alapjáték | 40 |
| Torpedó alap játék továbbfejlesztése..... | 45 |
| Torpedó alap játék továbbfejlesztése (haladóknak)..... | 45 |
| További alkalmak..... | 46 |



A :GAME ZIP 64 készlet bemutatása



A :GAME ZIP 64 osztálytermi készlet a következő eszközöket tartalmazza:

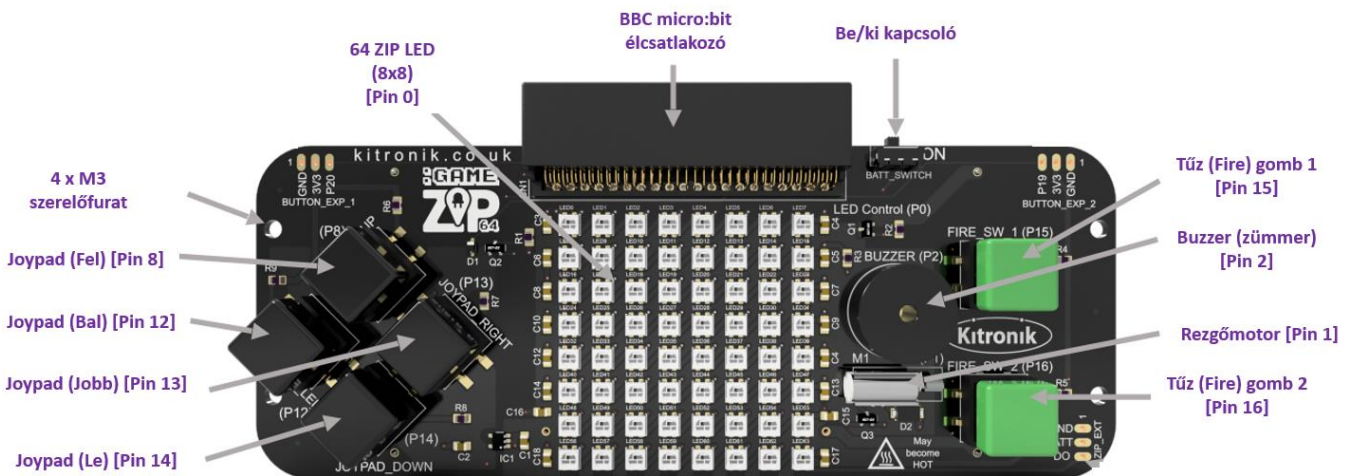
- 10 db :GAME ZIP 64 kontroller
- 10 db USB kábel
- 30 db AA elem

A kontrollerek működéséhez szükség van a micro:bit lapkákra is, azonban ezeket a készlet nem tartalmazza, így az iskolának már rendelkeznie kell a szükséges mennyiséggel.

A :GAME ZIP kontroller segítségével még látványosabb, élvezetesebb alkalmazásokat, játékprogramokat készíthetnek a diákok. A kontrolleren többek között 8×8-as RGB LED mátrix, 4 gomb az irányításhoz (fel, le, jobbra, balra), 2 tűz (fire) gomb, 1 db rezgőmotor és a hangok megszólaltatásához egy buzzer (zümmer) található.

Az eszköz felépítése

Az alábbi ábrán láthatjuk az eszköz főbb részeit. A micro:bit eszközt a felül található élcslakozóba kell bedugnunk.

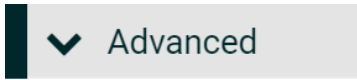


A 3 db AA elemet a hátoldalon található elemtartókban kell elhelyeznünk polaritásnak megfelelően.

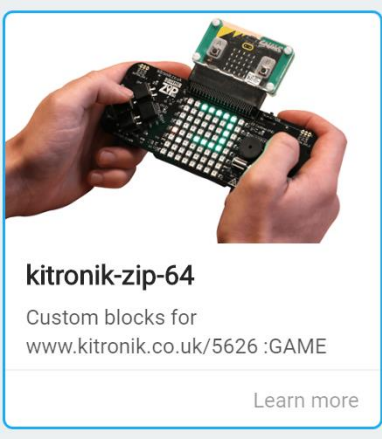
Programozás a makecode felületen

A Makecode felületet a <https://makecode.microbit.org/> webcímen találjuk.

Szükséges blokkok hozzáadása



+ Extensions



kitronik-zip-64
Custom blocks for
www.kitronik.co.uk/5626 :GAME
[Learn more](#)

A Makecode felülethez adjuk hozzá a Kitronik ZIP 64 nevű speciális blokkot. Ehhez kattintsunk az **Advanced** kategóriára, majd a **+Extensions** linkre.

Keressünk rá a „zip 64” kifejezésre. Ekkor megjelenik a **kitronik-zip-64** modul, amelyre kattintva telepíthetjük azt.

Ha a keresés nem adna találatot a kitronik ZIP 64-re, akkor a <https://github.com/KitronikLtd/pxt-kitronik-zip-64.git> webcímet be kell írni a mezőbe.

Ha mindent jól csináltunk, megjelenik a blokkok között

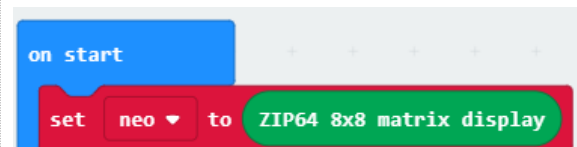
a következő kategória:  :GAME ZIP64

A :GAME ZIP64 kategóriában elérhető blokkok bemutatása

Display (Megjelenítő) szakasz





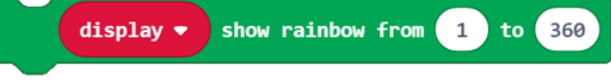
ZIP64 8x8 matrix display

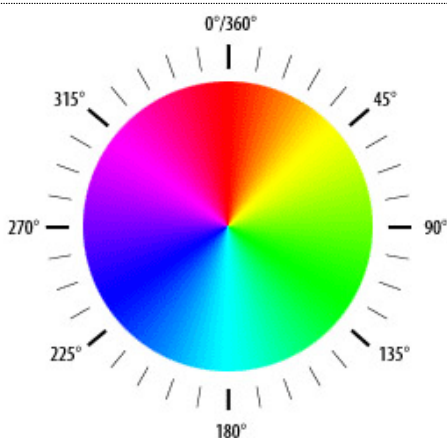
A LED mátrix inicializálásához szükséges. A blokkot egy változó értékének kell adnunk. A kijelző későbbi beállításainál majd erre a változóra hivatkozunk. pl.



string set matrix color at x 0 y 0 to red

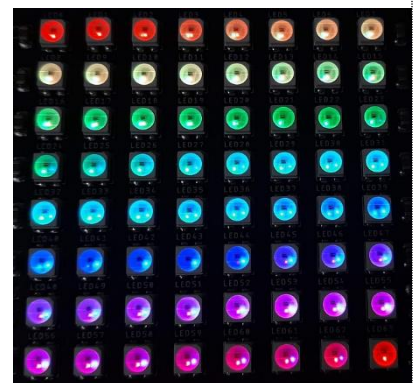
Az adott koordinátájú LED színének megváltoztatására szolgál. Az oszlopok és sorok nullától indexelődnek. A bal felső LED koordinátája: (0;0). A jobb alsóé: (7;7).


| | |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>A teljes LED mátrix színének beállítására szolgál. A választható színek: red, orange, yellow, green, blue, indigo, violet, purple, white, black.</p> |
|  | <p>A kijelző tartalmának megjelenítésére, frissítésére szolgál. Amennyiben pl. a LED mátrix elemeit koordinátáinként színeztük át, akkor szükség van a módosítások érvényre juttatására, amit ezzel a blokkal tehetünk meg.</p> |
|  | <p>A kijelzőt letörli, alaphelyzetbe állítja. De a változtatás akkor jut érvényre, ha utána a show blokkot is használjuk.</p> |
|  | <p>A kijelző fényerősségének beállítására szolgál (0-255). Vigyázzunk, mert a magas értékeknél nagyon vakító lehet a kijelző! Próbálkozzunk kicsi 10-20 körüli értékkel, és ha az nem elég, akkor növeljük meg,</p> |
|  | <p>A blokk segítségével a kijelző LED-jein a szivárvány színei jelennek meg. A megadott értékekkel azt tudjuk befolyásolni, hogy mi legyen a kezdő szín, illetve a végső szín.</p> |



A bal oldali színekörön láthatjuk, hogy melyik szögértéknek melyik szín felel meg.

Ha a szivárványt az alapbeállítás szerint rajzoljuk (1 és 360 között), akkor a jobb oldalon látható eredményt kapjuk.



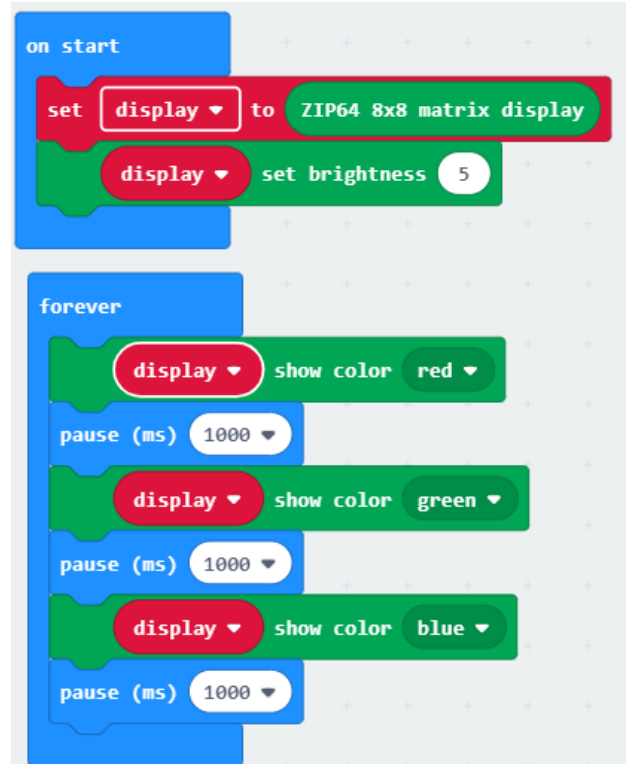
| | |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>A kijelzőn megjelenő pontok eltolására szolgál. Minden pont x koordinátája megnövekszik az itt beállított értékkel, vagyis jobbra tolódik. Ha így 63-nál nagyobb értéket kapnánk, akkor a 64-el vett maradék lesz az eredmény. Negatív értéket is be lehet állítani, ekkor a pontok balra tolódnak. Ha a sorokat akarjuk eltolni, akkor nyilván ± 8-as paramétert kell használni.</p> |
|  | <p>Színkonstans, amely a színek beállítására szolgál. A választható színek: red, orange, yellow, green, blue, indigo, violet, purple, white, black.</p> |
|  | <p>A blokk segítségével tetszőleges színt beállíthatunk, az RGB koordináta-rendszer szerint.</p> |
| Input (Bemenet) szakasz | |
|  | <p>Logikai feltételeknél használhatjuk. A kifejezés igaz értéket ad vissza, ha a Joypad vagy Fire (tűz) gombok le lettek-e nyomva.</p> |
|  | <p>Ezzel adhatjuk meg, hogy mi történjen (milyen blokkok hajtódnak végre) a Joypad és Fire (tűz) gombok megnyomásakor.</p> |
| Feedback (visszajelzés) szakasz | |
|  | <p>Ezzel a blokkal a rezgőmotort kapcsolhatjuk be a megadott ideig. Ilyenkor a kontroller el kezd rezegni.</p> |
|  | <p>Amennyiben a hangokat a kontroller zümmerjén akarjuk megszólaltatni, ezt a blokkot kell használnunk a projektünk elején.</p> |

Egyszerű demó projektek a korábban ismertetett blokkok megismeréséhez

1. Színek váltakozása

Az alábbi projektben beállítjuk a fényerőt 5-ös értékre és a piros, kék, zöld színeket váltogatjuk a teljes kijelzőn 1 másodperces késleltetéssel.

A projekt elérhetősége:
https://makecode.microbit.org/_C1aid76JWipy

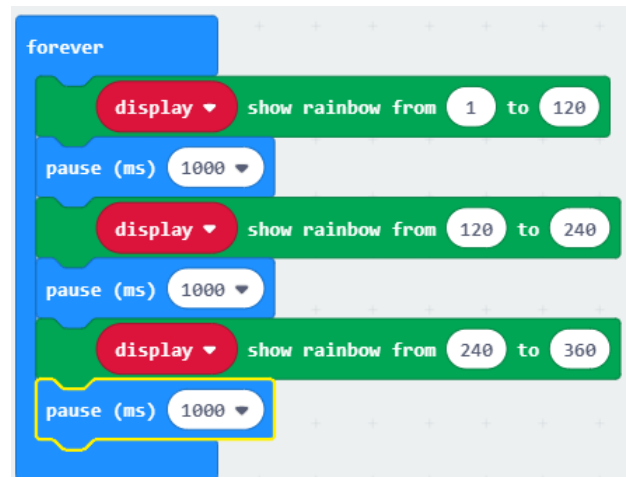
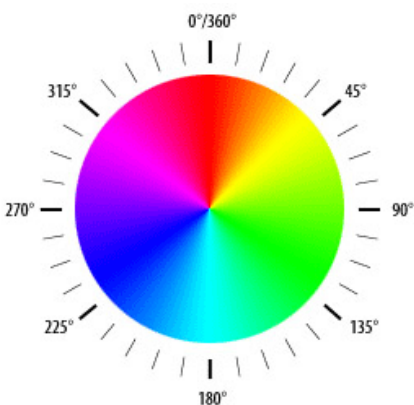


```
on start
  set display to ZIP64 8x8 matrix display
  display set brightness 5

forever
  display show color red
  pause (ms) 1000
  display show color green
  pause (ms) 1000
  display show color blue
  pause (ms) 1000
```

2. Szivárvány színeinek megjelenítése

Ebben a projektben a szivárvány színeit jelenítjük meg, mégpedig három fázisban.



```
forever
  display show rainbow from 1 to 120
  pause (ms) 1000
  display show rainbow from 120 to 240
  pause (ms) 1000
  display show rainbow from 240 to 360
  pause (ms) 1000
```

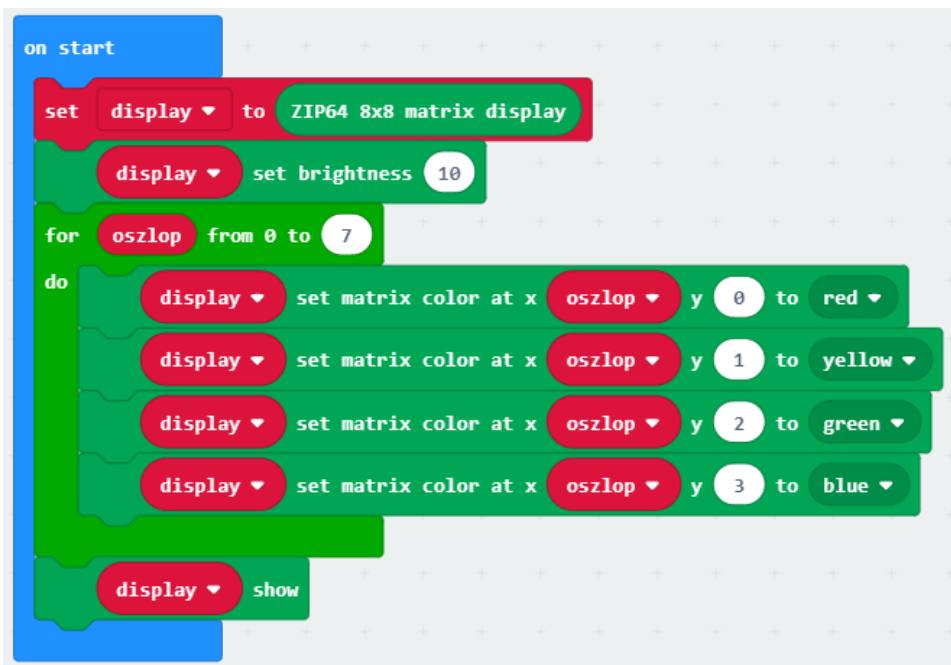
Az On start blokkunk tartalma ugyanaz, mint az előbbi esetben, ezért azt nem tüntettük fel.

Először 1 és 120 között (piros-zöld), aztán 120 és 240 (zöld-kék) között, majd 240 és 360 (kék-piros) között.

A projekt elérhetősége:
https://makecode.microbit.org/_goVTY7YdTK3D

3. LED-ek színeinek beállítása koordinátánként

Ebben a projektben soronként más-más színt állítunk be a LED mátrixon. Ezt úgy oldjuk meg, hogy a megfelelő koordinátájú pontokat átszínezzük. A LED mátrix 1. sora piros, a 2. sárga, a 3. zöld, a 4. kék lesz. A koordináták 0-tól indexelődnek, így az 1. sor/oszlop a 0-s indexű, a 8. sor/oszlop pedig a 7-es indexű.

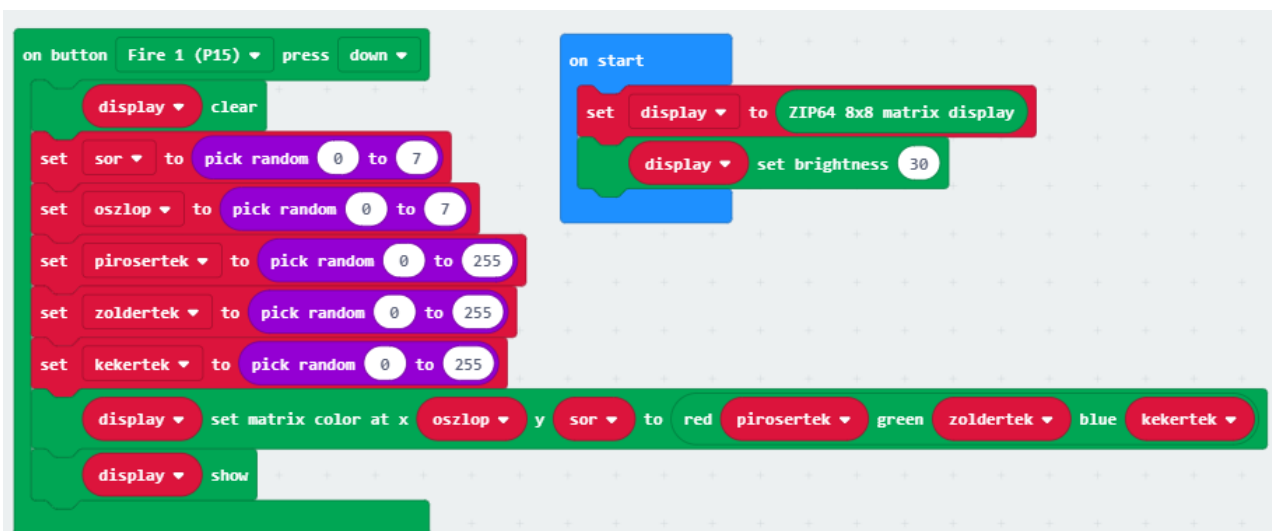


```
on start
  set display to ZIP64 8x8 matrix display
  display set brightness 10
  for oszlop from 0 to 7
  do
    display set matrix color at x oszlop y 0 to red
    display set matrix color at x oszlop y 1 to yellow
    display set matrix color at x oszlop y 2 to green
    display set matrix color at x oszlop y 3 to blue
  display show
```

Amennyiben koordinátánként színezzük át a LED-eket, fontos, hogy a kirajzolás után a **Show** blokkot is helyezzük el, különben nem jelenik meg az eredmény.

A projekt elérhetősége: https://makecode.microbit.org/_DpTUwa5efYoH

4. Véletlenszerűen kiválasztott LED-ek, véletlenszerű színnel



```
on button Fire 1 (P15) press down
  display clear
  set sor to pick random 0 to 7
  set oszlop to pick random 0 to 7
  set pirosertek to pick random 0 to 255
  set zoldertek to pick random 0 to 255
  set kekertek to pick random 0 to 255
  display set matrix color at x oszlop y sor to red pirosertek green zoldertek blue kekertek
  display show

on start
  set display to ZIP64 8x8 matrix display
  display set brightness 30
```

A fenti projektben a véletlenszerűen kiválasztott LED-eket átszínezzük a véletlenszerűen meghatározott színekre. A színt RGB koordináta-rendszer szerint adjuk meg. A vörös, zöld és kék komponens is 0 és 255 között fog egy véletlenszerűen meghatározott értéket felvenni.

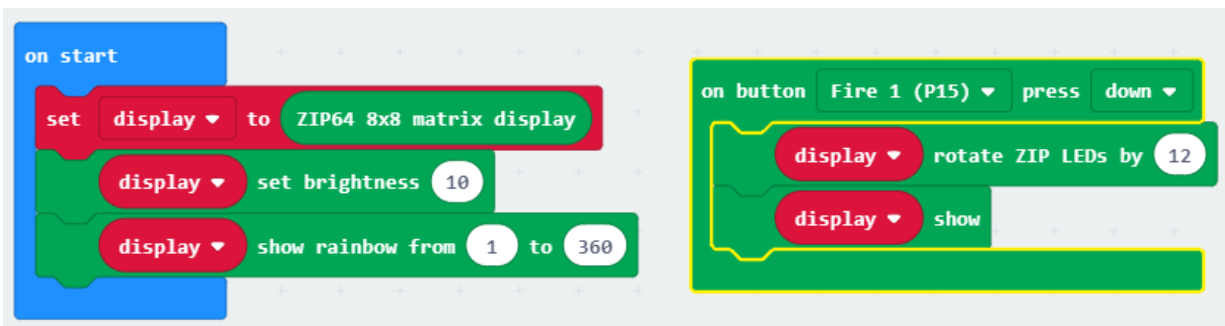
A megjelenítés a Tűz 1 (Fire 1) gomb megnyomásakor történik meg.

A projekt elérhetősége: https://makecode.microbit.org/_1FEUpH4d2ex9

5. Pontok eltolása

A LED mátrixon megjelenő színeket akár el is tolhatjuk egy megadott értékkel.

Ebben a projektben megjelenítjük a szivárvány színeit, majd a Tűz1 (Fire 1) gomb hatására 12 lépéssel eltoljuk jobbra, így a színek minden gombnyomásnál meg fognak változni (minden pont jobbra tolódik 12 egységgel).

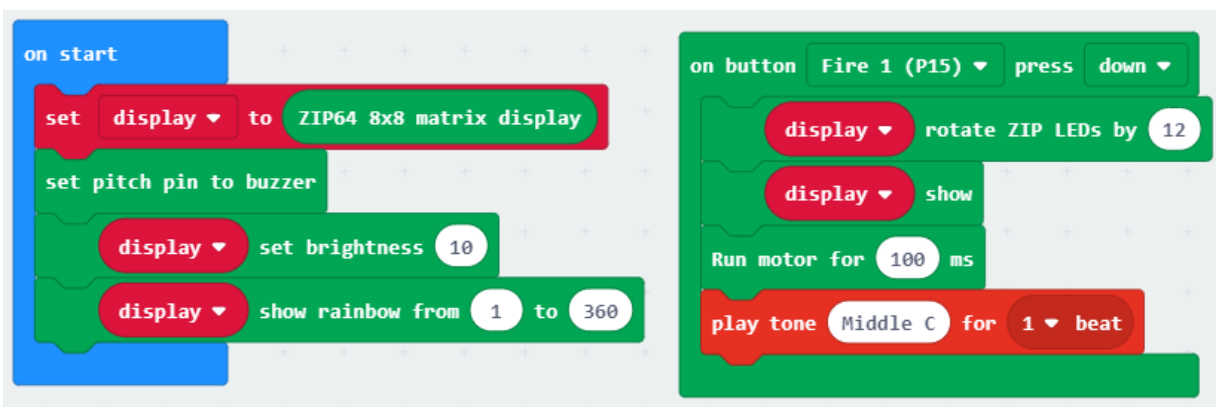


A projekt elérhetősége: https://makecode.microbit.org/_W8kaYP5dphDK

6. Rezgőmotor és zümmer használata

Az előző projektet továbbfejlesztjük úgy, hogy a Tűz 1 gomb megnyomásakor lejátszunk egy hangot, és a rezgőmotort is elindítjuk.

Ahhoz, hogy a hang a zümmeren játszódjon le, a `set pitch pin to buzzer` blokkot el kell helyezni a projektünkben.



A projekt elérhetősége: https://makecode.microbit.org/_P7fcFkLbmb3D

A szakkör felépítése

| | |
|--------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alkalmak száma | 6 |
| Óraszám/alkalom | 2 * 45 perc = 90 perc |
| Ajánlott korosztály | 10 – 16 év |
| Eszközsükséglet 15 diákra + 1 tanárra vetítve | <ul style="list-style-type: none">– 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 GAME:ZIP kontrolller– 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 akár hozzászólhasson. |

Szakköri anyagunk célcsoportját azon diákok jelentik, akik már a micro:bit eszköz alapvető programozásában jártasságot szereztek. Ezzel kapcsolatos szakköri anyagunk a <http://microbit.inf.elte.hu/szakkori-anyag/> címen nyilvánosan elérhető.

Könyvünkben már a korábban megszerzett ismeretekre építünk, kihasználva a :GAME ZIP 64 controllerben rejlő lehetőségeket. Ennek során továbbra is a MakeCode blokkprogramozási felületet (<https://makecode.microbit.org>) használjuk.

Jelen kiadványunk informatika tanárok számára készült, ezért nem térünk ki az alapvető programozással kapcsolatos fogalmak ismertetésére (pl. ciklus, elágazás). A bemutatott tematika csak egy sorvezető, ettől szabadon el lehet térni, sőt arra biztatunk 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 a micro:bit lapkában rejlő lehetőségeket egy új eszköz bevonásával kiterjesszük, hogy új kihívások elé állítsuk a diákokat.

A kiadványunkban bemutatott példák, alkalmazások mellett a <https://www.kitronik.co.uk/blog/lesson-plans-game-zip-64-microbit/> oldalon található óratervek, 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. Ismerkedés az eszközzel
2. Játék a színekkel
3. Irány az űr! (űrhajós játék)
4. Kapj el! (alap játék, amelyre sok más játék épül)
5. Kígyó játék
6. Torpedó játék
7. Ötletek a további alkalmakra

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 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 (Ismerkedés az eszközzel)

| Tematikai egység | Alkalmazott módszerek, munkaformák | Időtartam (perc) |
|----------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 1. Az eszköz bemutatása, munkavédelmi előírások | Frontális tanári bemutató | 10 perc |
| 2. A :GAME ZIP 64 lehetőségei a gyakorlatban | Közös alkalmazásfejlesztés, magyarázattal | 25 perc |
| 3. Közlekedési lámpa szimuláció készítése | Egyéni munka | 25 perc |
| 4. Közlekedési lámpa szimuláció egyéni, vagy páros továbbfejlesztése | Egyéni, vagy páros munka | 20 perc |
| 5. Az elkészült munkák megtekintése | A diákok bemutatják egymásnak az elkészült munkákat. Közösen megbeszéljük az egyes munkák pozitívumait, és a lehetséges továbbfejlesztési ötleteket. | 10 perc |

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!
- Mindig tiszta felületen dolgozzunk, ahol nincsenek olyan tárgyak, amelyek rövidzárlatot okozhatnak (tűzőgép kapocs, gemkapocs)!



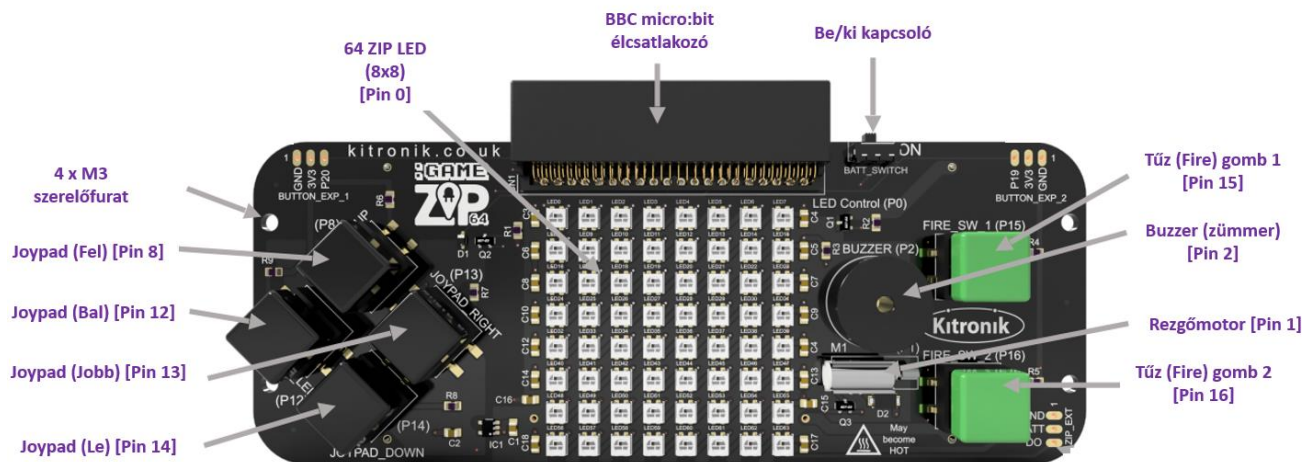
Open page



- Mielőtt az eszközhöz hozzáérnénk, 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 munkakörnyezetet úgy alakítsuk ki, hogy ne eshessen le, illetve véletlenül se lehessen lesodorni a micro:bitet!
- A különböző gesztusok kipróbálásakor (pl. rázás, eldöntés stb.) biztosan tartsuk az eszközt a kezünkben, nehogy véletlenül leejtsük.
- A GAME:ZIP kontroller működéséhez az AA elemeket el kell helyezni az elemtartóban a polaritásuknak megfelelően. USB kábellel nem működik az eszköz! Ne használjunk Li-Ion vagy Li-polymer cellákat, akkor sem, ha hasonló csatlakozókkal is szerelték őket, mert azok a termék tönkremeneteléhez vezethetnek!
- 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 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.
- A kontrollerek épségére vigyázzunk! A habszivacs tartóból óvatosan vegyük ki, mivel a kiálló alkatrészek (különösen az elemtartók) sérülékenyek!
- Az elemek cseréjénél szintén kellő gondossággal járjunk el! Vigyázzunk, hogy a csere során ne sérüljenek meg az elemtartók és az alkatrészek!
- Fontos! A Neopixel LED mátrix eléggé vakítóan tud világítani. A szemek és az elemek kímélése érdekében mindig alacsony világosságértéket (brightness) használjunk! (pl. 5, a maximum 255 helyett) a programozás során.
- A LED-ek huzamosabb használat és nagy fényerő mellett felforrósodhatnak, így azokat lehetőleg ne érintsük meg!
- Amikor nem használjuk az eszközt, az On/Off kapcsolóval kapcsoljuk ki, különben az elemek gyorsan lemerülnek.

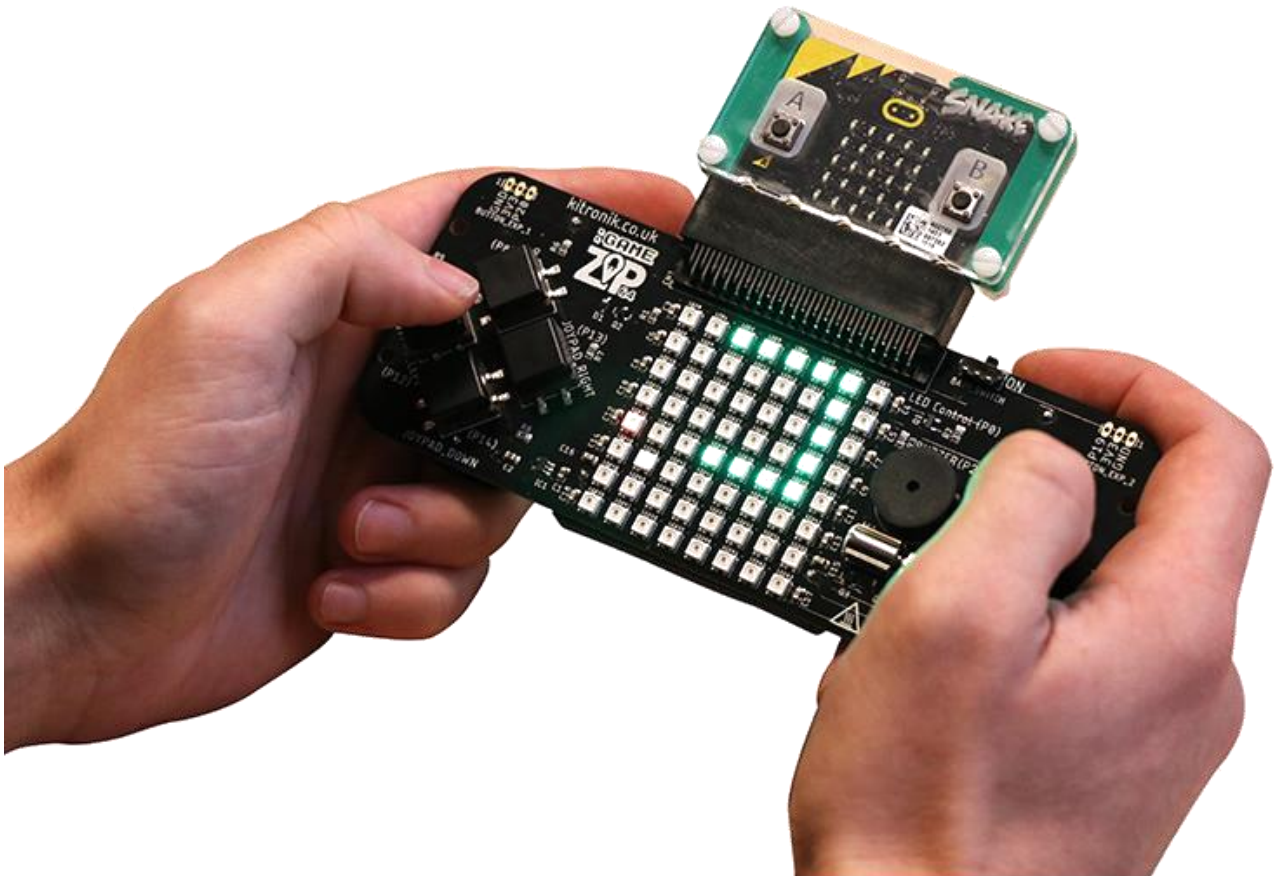
Az eszköz bemutatása

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



Térjünk ki arra, hogy a micro:bithez képest melyek az újdonságok, amelyeket az alkalmazások fejlesztése során kihasználhatunk:

- Színes, 8x8-as LED kijelző
- Joypad (fel, le, jobbra, balra gombokkal)
- Tűz 1 és Tűz 2 gombok
- Rezgőmotor (rezgő visszajelzéshez)
- Zümmer (hangok megszólaltatására)



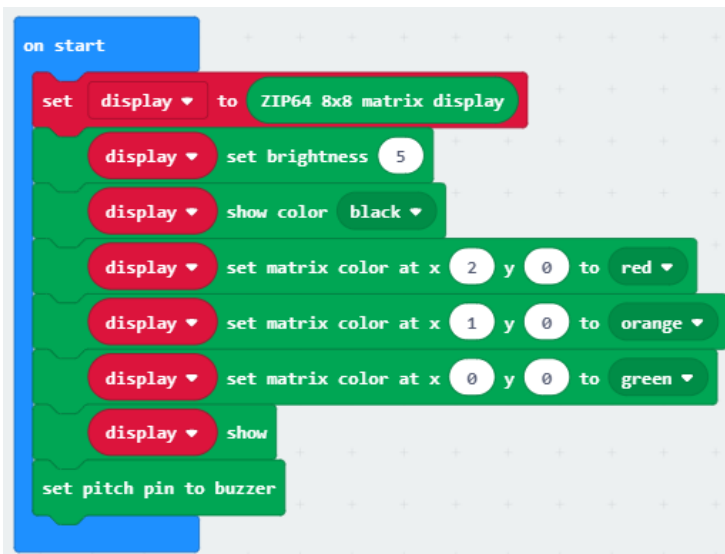
A :GAME ZIP 64 lehetőségei a gyakorlatban

A következő demo projektet készítsük el közösen. A projekt segítségével bemutatjuk az újonnan elérhető lehetőségeket.

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_3FbTE5g1Fhae

A demó alkalmazásban egy zöld, sárga és piros színű LED-eket fogunk a 4 irányba eltolni, illetve megjelenítünk egy szivárvány mintát. Ezzel minden olyan blokkot megmutatunk a gyakorlatban, amelyekre később építhetünk a fejlesztéseknél.

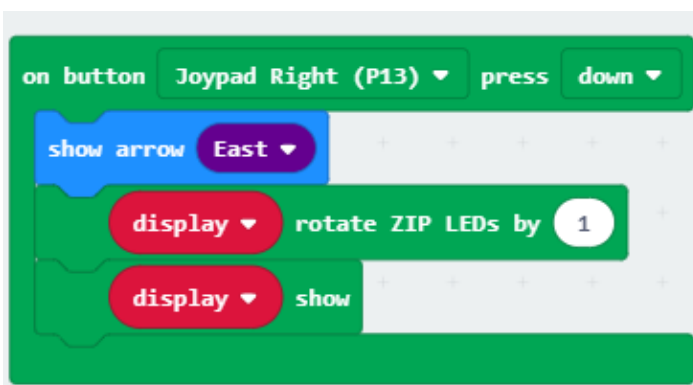


Először inicializáljuk a kijelzőt, beállítjuk a világosság szintjét, majd a háttérszínt feketére állítjuk (a törlés (clear) is megfelelő lett volna).

Ezek után kirajzoljuk egymás mellé a piros, narancs, zöld pontokat.

Ahhoz, hogy érvényre jussanak a változások a kijelzőn lévő változásokat meg kell jeleníteni (**show**).

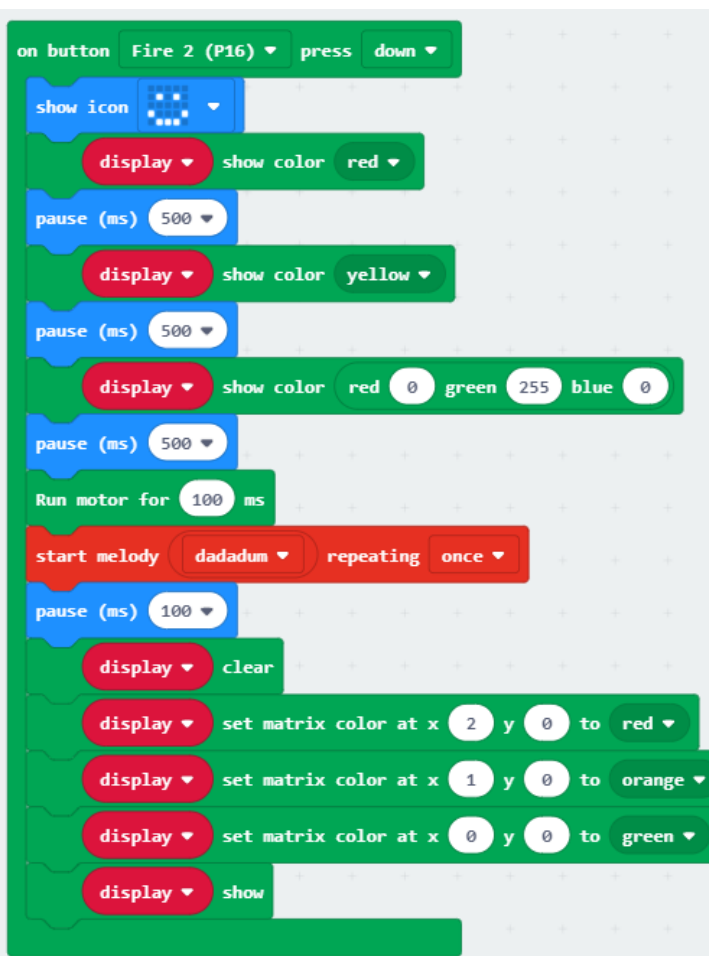
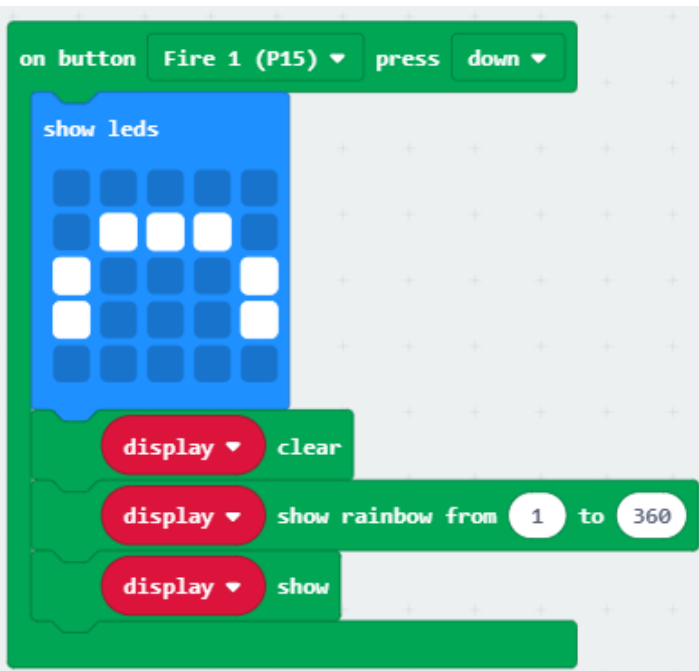
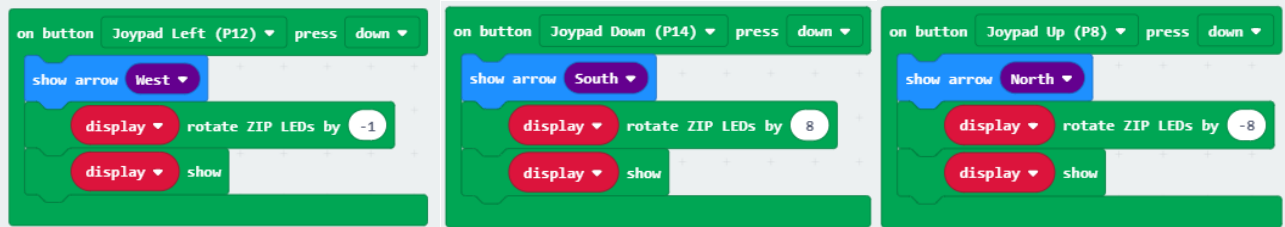
Végül pedig beállítjuk, hogy a hangokat a zümmerre szeretnénk irányítani.



Folytassuk azzal, hogy a Joypad jobb gombjának lenyomásakor legyen eltolva jobbra a kirajzolt ábra.

Ezzel párhuzamosan a micro:bit kijelzőjén jelenjen meg a jobbra (Kelet felé) mutató nyíl, jelezve az eltolás irányát.

Hasonlóképpen készítsük el a többi blokkot is. A lefele és felfele történő eltolásnál már 8-as értéket kell használnunk. A controlleren az egyes LED-ek apró betűvel meg vannak sorszámozva, ami segítség lehet a koordináták meghatározásánál.



Most azzal folytatjuk, hogy a A Tűz 1 gomb hatására a micro:bit kijelzőjén jelenjen meg egy szivárványra emlékeztető (félkör) ikon.

A :GAME ZIP kijelzőjét pedig töröljük le, majd rajzoljunk ki egy szivárvány mintát.

A Tűz 2 gomb hatására jelenítsünk meg a micro:bit kijelzőjén egy vigyorgó fejet. A :GAME ZIP kijelzőjén pedig felváltva a piros, sárga és zöld színek jelenjenek meg kis késleltetéssel.

A zöld szín kódját az RGB színkoordinátarendszer szerint adjuk meg, hogy erre is lássanak példát a diákok.

Utána működésbe hozzuk a rezgőmotort, majd lejátszunk egy kis dallamot.

Utána letöröljük a kijelzőt, és visszaállítjuk az eredeti állapotot a három pont megjelenítésével, és frissítjük a kijelzőt.

Közlekedési lámpa szimuláció készítése

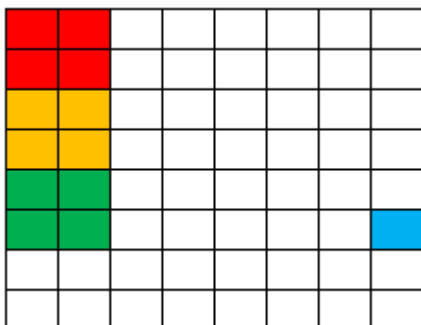


Feladat a diákok számára



Készítsetek egy közlekedési lámpa szimulációt!

- A kontroller kijelzőjén jelenjen meg a piros, sárga, zöld közlekedési lámpa. Ne csak 1 LED legyen kigyújtva, hanem 4 (kettő-kettő egymás alatt és mellett, ahogy az alábbi képen is látszik)



- Az utolsó oszlopban a kék színű pont azonosítja az autó helyzetét. Ennek a pontnak a közlekedési lámpa szerint kell haladnia a kijelzőn fentről lefele. Ha pirosra vált a lámpa, meg kell állnia. Zöld jelzésnél el kell indulnia.
- A lámpák időzítve váltsanak piros, piros-sárga, zöld, sárga, piros sorrendben.
- A micro:bit kijelzőjén jelenjen meg egy visszaszámláló, amely mutatja, hogy mikor fog váltani a lámpa pirosról, illetve zöldről.

Közlekedési lámpa szimuláció egyéni továbbfejlesztése

Adjunk teret az alkalmazás egyéni továbbfejlesztésének.

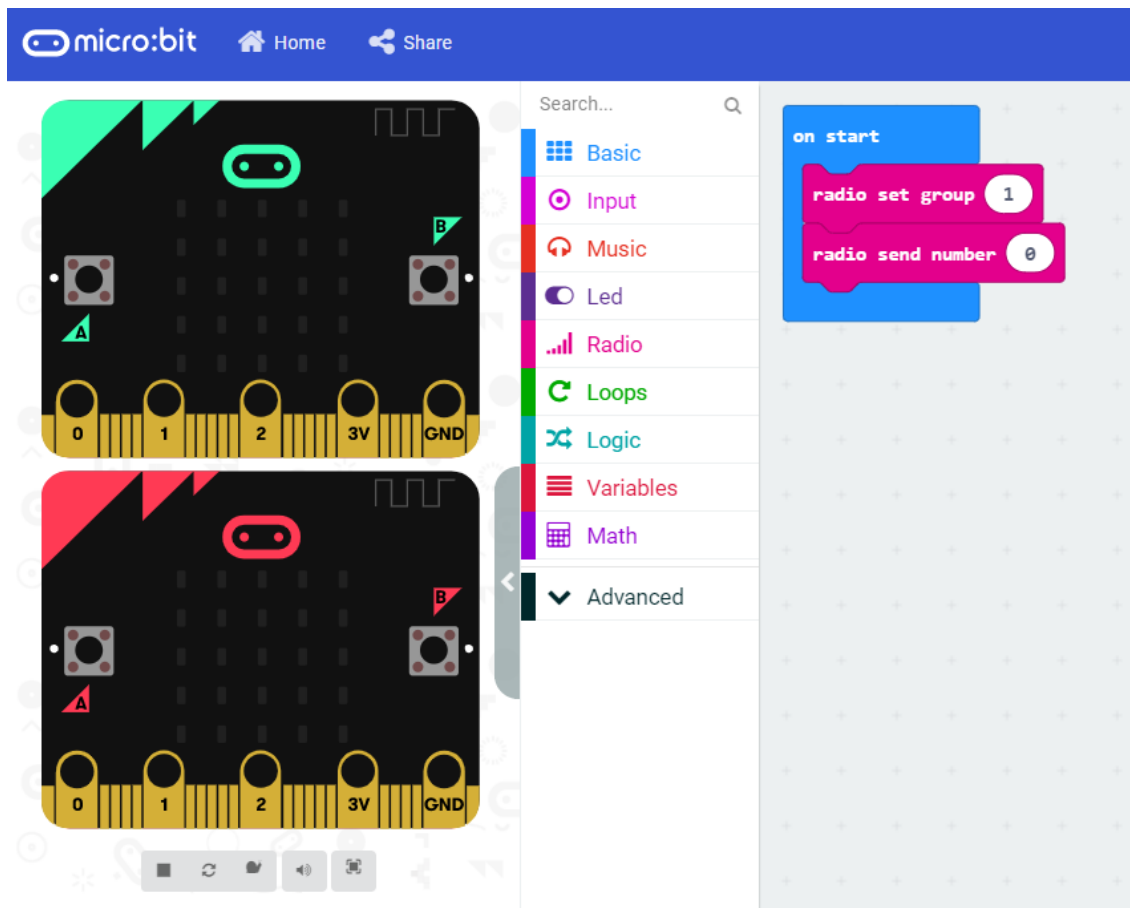
Néhány ötlet:

- Ne csak visszaszámláló legyen, hanem hangjelzés is hívja fel a figyelmet a lámpaváltásokra.
- Ne csak 1 autó legyen az utolsó oszlopban, hanem több is, sőt akár egymás mellett sávok is kialakíthatók.
- Lehesse manuálisan is váltani a lámpák állapotai között a Tűz 2 gomb lenyomásával. Ekkor a visszaszámlálás a gomb megnyomásakor induljon el.

Közlekedési lámpa szimuláció páros továbbfejlesztése (haladóknak)

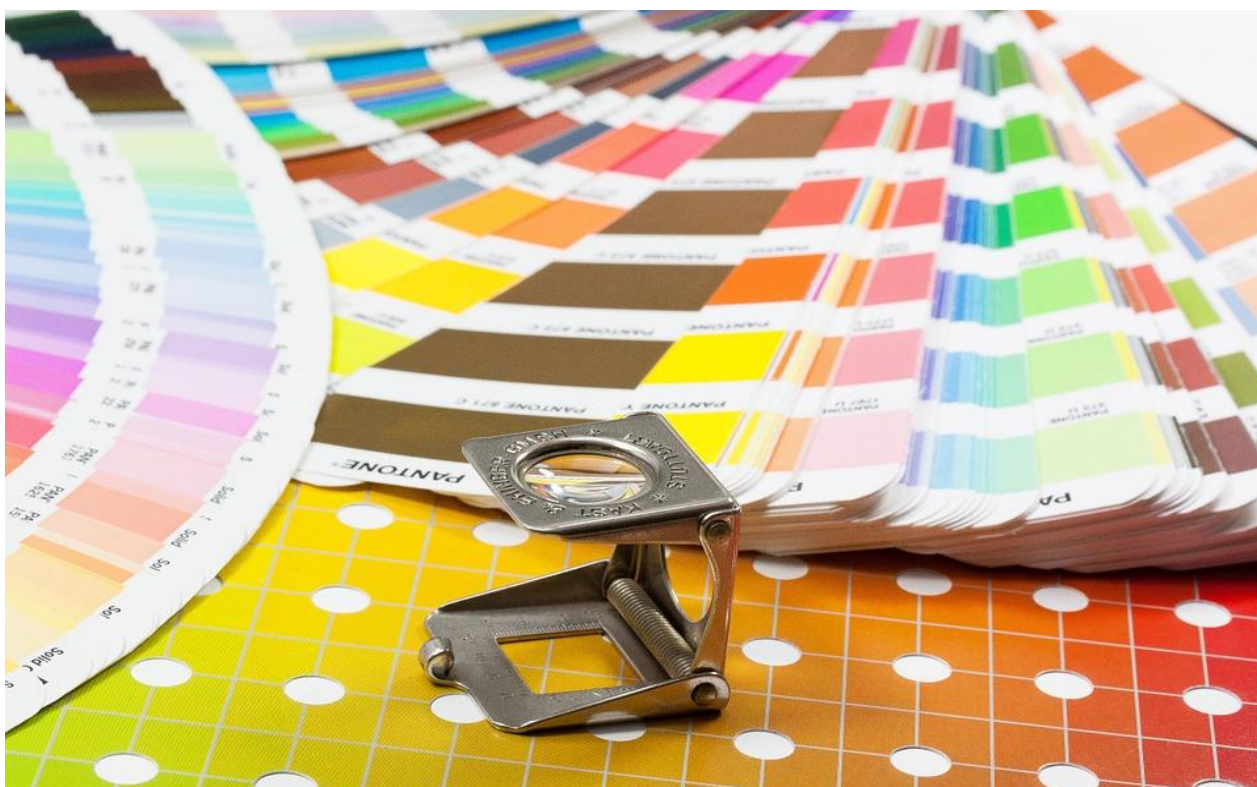
Feladat a diákok számára (haladóknak)

A micro:bitek képesek egymással kommunikálni rádió kapcsolaton keresztül. Készítsetek olyan alkalmazást, ahol az egyik controller a közlekedési lámpát valósítja meg, a másik controlleren pedig az autók a lámpa jelzései alapján közlekednek (megállnak, ha piros a lámpa, zöld, ha elindulnak).



2. alkalom (Játék a színekkel)

| Tematikai egység | Alkalmazott módszerek, munkaformák | Időtartam (perc) |
|------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 1. Játék a színekkel | Közös alkalmazásfejlesztés, magyarázattal | 30 perc |
| 2. Játék a színekkel - továbbfejlesztés | Egyéni munka | 15 perc |
| 3. Játék a színekkel – saját játék fejlesztése pármunkában | Ötletelés és alkalmazásfejlesztés páros munkában | 35 perc |
| 4. Az elkészült munkák megtekintése. | A diákok bemutatják egymásnak az elkészült játékokat, amelyeket ki is lehet próbálni. Közösén megbeszéljük az egyes munkák pozitívumait, és a lehetséges továbbfejlesztési ötleteket. | 10 perc |

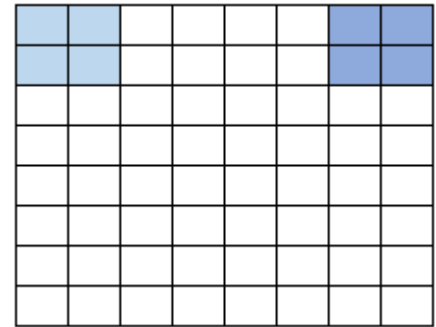


Játék a színekkel

A tanár által elkészítendő/bemutatandó mintaprojekt

<https://makecode.microbit.org/VJpEkaXCtFuX>

Most egy egyszerű játékot fogunk készíteni, részben ismételve az előző alkalommal megismertetett lehetőségeket.



A játék lényege, hogy a :GAME ZIP kijelzőjének bal és jobb oldalán is megjelenítünk adott színű blokkokat, amelyek vagy azonosak, vagy csak kis mértékben térnek el a színükben. A játékos feladata, hogy megtippelje, hogy a két szín különbözik-e, vagy sem.

Minden tipp után kijelezzük, hogy jó volt-e a játékos tippje (egy vigyorgó, vagy szomorú fej megjelenítésével a micro:bit kijelzőjén), a játék végén pedig megjelenítjük a helyes tippek számát, vagyis az elért pontszámot.

A játékban 10 színpárosítást kell megtippelni. A válaszadásra 4 másodperc áll rendelkezésre. A Tűz 1 gomb jelenti azt, hogy a játékos szerint a két szín különbözik, a Tűz 2 pedig azt, hogy a két szín azonos.

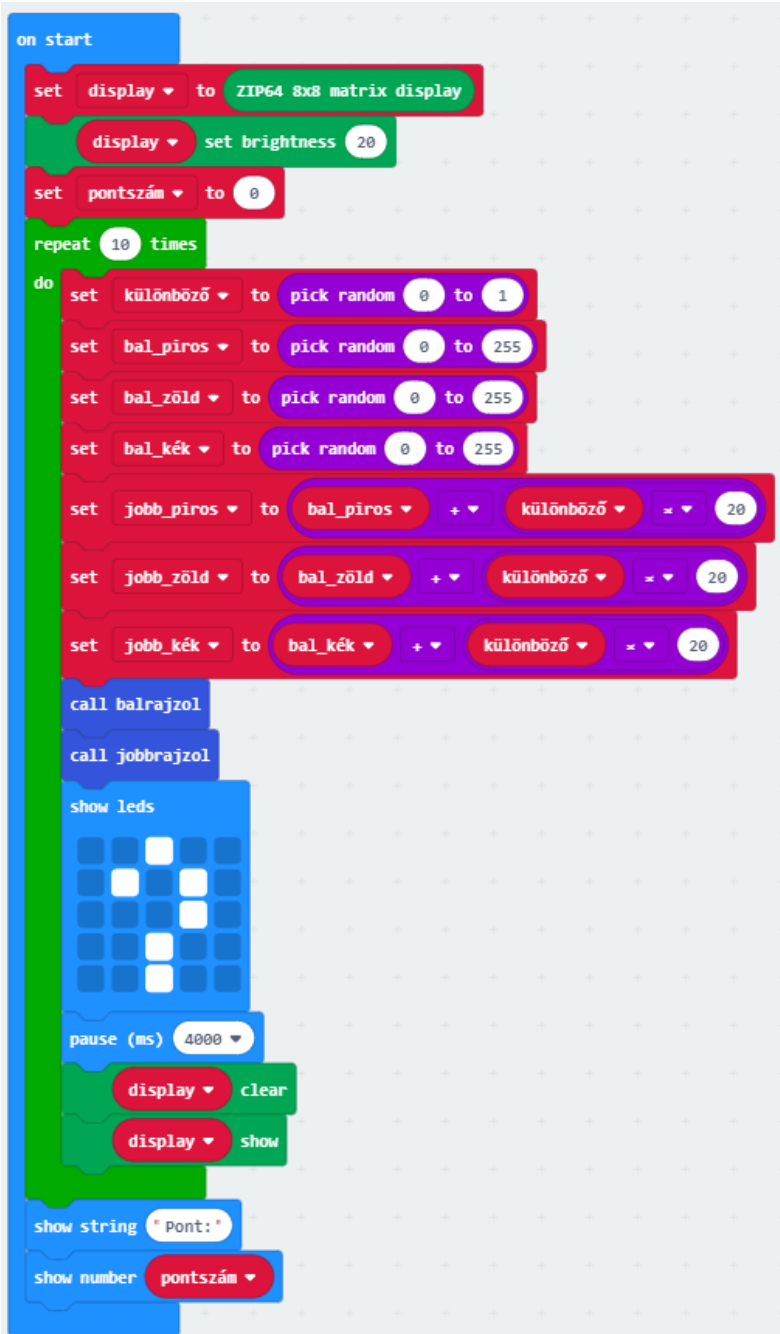
Az alábbiakban az **on start** blokkot láthatjuk, alatta pedig a magyarázatot. Amikor az alkalmazást közösen készítjük természetesen az egyes blokkok elhelyezésének sorrendjétől eltérhetünk. (pl. később térünk ki a pontszám kezelésére, nem mindjárt az elején hívjuk fel erre a figyelmet.)

A játék indulásakor inicializáljuk a kijelzőt és beállítjuk a világosság szintjét 20-as értékre.

Beállítjuk a pontszám alapértékét nullára. Ezen változó értékét fogjuk növelni később, amikor a játékos helyesen találta el, hogy azonosak vagy különbözőek-e a színek.

A játékban összesen 10 alkalommal kell tippelnünk, ezért a játék működtetésért felelős blokkokat egy ciklusban helyezzük el (**repeat 10**).

A bal oldalon megjelenő színblokk színeit véletlenszerűen határozzuk meg, vagyis a RGB komponenseket 0 és 255 közti véletlenszámmal töltjük fel. Az eredmény a *bal_piros*, *bal_zöld*, *bal_kék* változókba kerülnek.



Szintén véletlenszám dönti el, hogy a jobb oldali színblokk különböző legyen-e a bal oldalitól, vagy azonos vele. Itt akár igaz/hamis értéket is választhatnánk, de ebben az esetben praktikusabb 0 és 1 számokból választani véletlenszerűen. A 0 jelenti, hogy azonos, 1 pedig azt, hogy különböző színű lesz a két blokk.

Ezután meghatározzuk a jobb oldali színblokk színeit is (*jobb_piros, jobb_zöld, jobb_kék*) mégpedig úgy, hogy ha azonosnak kell lennie, akkor ugyanaz lesz az érték, mint a bal oldali blokk esetén. Ha pedig különböznie kell, akkor 20-al megnöveljük mindegyik RGB komponens értékét.

Itt látszik, hogy miért volt jó választás a 0 és 1, hiszen ezzel az értékkel szorozva a 20-as értéket

vagy nullát, vagy 20-at kapunk, vagyis azonos, vagy növelt eredményt kapunk.

Hogy átláthatóbb legyen a kódunk, a színblokkok kirajzolását érdemes függvényekbe szervezni (*balrajzol, jobbrajzol*) és ezeket meghívni.

A micro:bit kijelzőjén megjelenítünk egy kérdőjelet, és 4 másodpercig várunk mielőtt letöröljük a képernyőt.

A ciklus után elhelyezett blokkok felelnek a pontszám kiírásáért a micro:bit kijelzőjén.

A bal és jobb oldali blokkok kirajzolása hasonlóan történik az alábbiak szerint:

```
function balrajzol
  display set matrix color at x 0 y 0 to red bal_piros green bal_zöld blue bal_kék
  display set matrix color at x 1 y 0 to red bal_piros green bal_zöld blue bal_kék
  display set matrix color at x 0 y 1 to red bal_piros green bal_zöld blue bal_kék
  display set matrix color at x 1 y 1 to red bal_piros green bal_zöld blue bal_kék
  display show
```

```
function jobbrajzol
  display set matrix color at x 6 y 0 to red jobb_piros green jobb_zöld blue jobb_kék
  display set matrix color at x 7 y 0 to red jobb_piros green jobb_zöld blue jobb_kék
  display set matrix color at x 6 y 1 to red jobb_piros green jobb_zöld blue jobb_kék
  display set matrix color at x 7 y 1 to red jobb_piros green jobb_zöld blue jobb_kék
  display show
```

A Tűz 1 gomb megnyomásával azt jelzi a játékos, hogy a két színblokk szerinte különbözik. Ha ez teljesül, akkor egy vigyorgó fejet rajzolunk ki és megnöveljük a pontszámot, ellenkező esetben egy szomorú fej jelenik meg.

A Tűz 2 gomb nagyon hasonló, csak itt akkor tippelt helyesen a felhasználó, ha a két szín azonos, vagy nem különböző (*not különböző*)

```
on button Fire 1 (P15) press down
  if különbözõ then
    show icon [smiling face]
    change pontszám by 1
  else
    show icon [sad face]
```

```
on button Fire 2 (P16) press down
  if not különbözõ then
    show icon [smiling face]
    change pontszám by 1
  else
    show icon [sad face]
```


Játék a színekkel - továbbfejlesztés

Feladat a diákok számára

Fejlesszék tovább az alkalmazást az alábbiak szerint

- A játékos válaszána helyessége alapján egy vidám, vagy szomorú dallam is játszódjon le.
- Rossz válasz esetén legyen rezgőmotoros visszajelzés is.
- Most 20-as értéket adtunk hozzá mindegyik színértékhez. Legyen ez is véletlenszám, 15 és 25 között, és ne csak növeljük az értéket, hanem akár ennyivel csökkenhessen is.
- Ne 2x2-es blokk jelenjen meg, hanem nagyobb.

Játék a színekkel – saját játék fejlesztése pármunkában

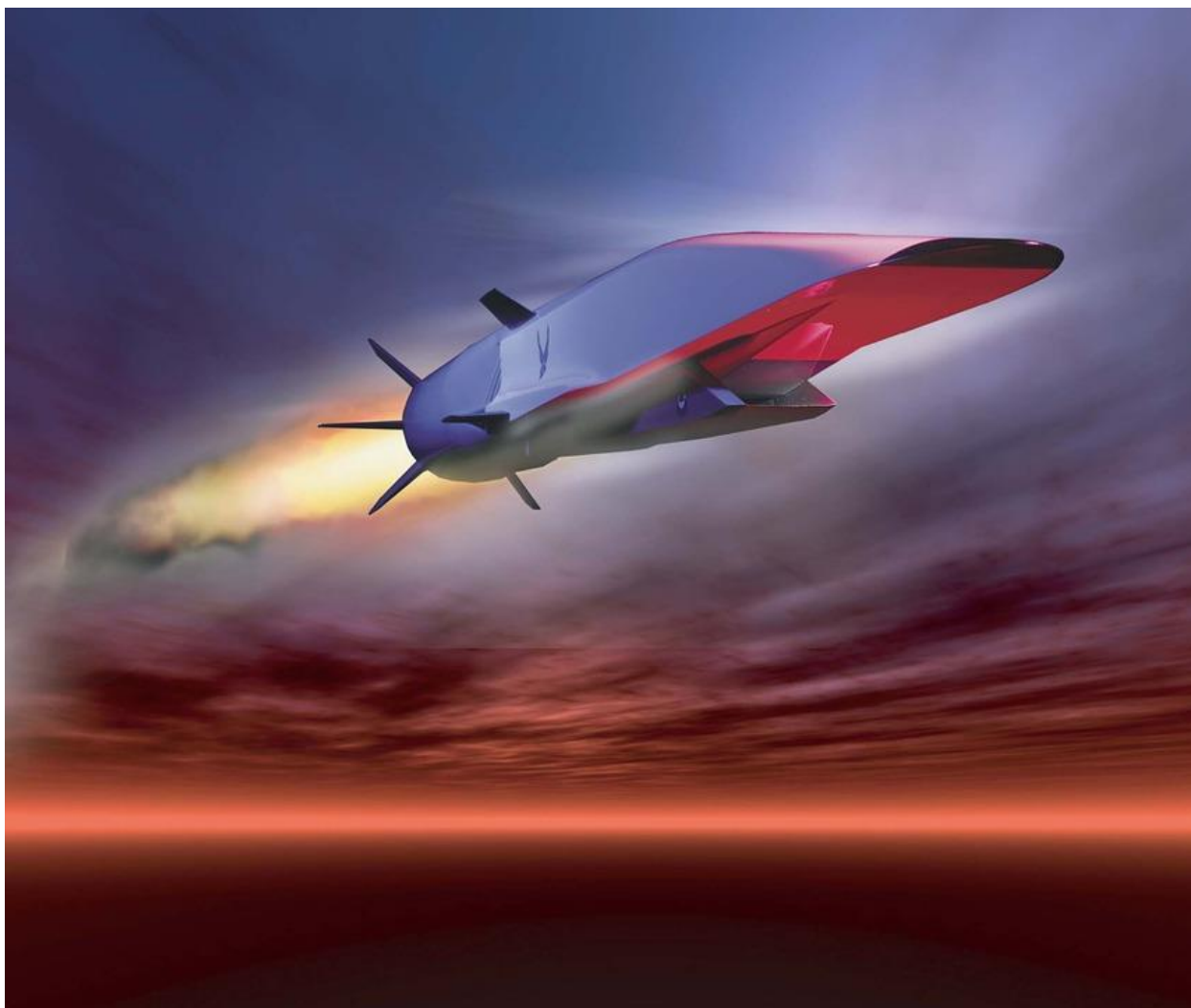
Feladat a diákok számára

Találjatok ki olyan játékot, amely a színérzékelésre, vagy világosságérzékelésre alapszik és valósítsátok meg a rendelkezésre álló időben.

Mutassátok be az elkészült játékokat, és próbáljátok ki Ti is mások játékait!

3. alkalom (Irány az űr!)

| Tematikai egység | Alkalmazott munkafarmák | módszerek, | Időtartam (perc) |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------------|
| 1. Űrhajós játék | Közös alkalmazásfejlesztés, magyarázattal | | 30 perc |
| 2. Űrhajós játék - továbbfejlesztés | Ötletelés, egyéni illetve páros munka | | 45 perc |
| 3. Az elkészült munkák megtekintése. | A diákok bemutatják egymásnak az elkészült játékokat, amelyeket ki is lehet próbálni. Közösen megbeszéljük az egyes munkák pozitívumait, és a lehetséges továbbfejlesztési ötleteket. | | 15 perc |



Úrhajós játék

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_1xAJqzVkoVzw

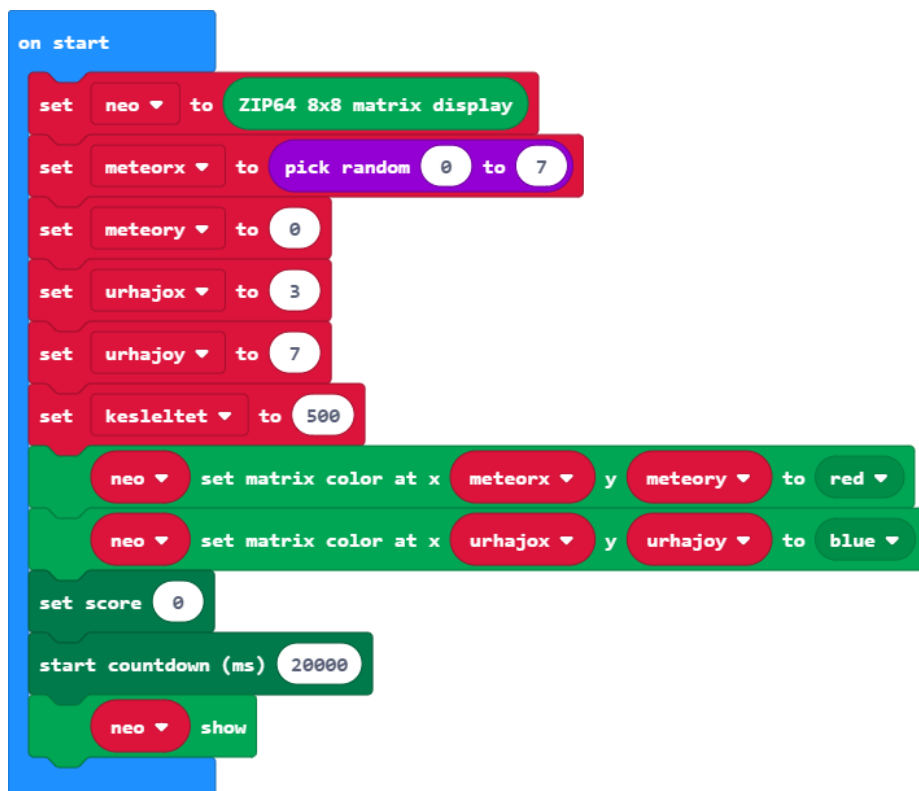
A szakköri anyagunk (<http://microbit.inf.elte.hu/szakkori-anyag/>) 65. oldalán ismertetett úrhajós játékot valósítjuk meg az eszközre hangolva.

A játék lényege: A kijelző alsó sorában jelenjen meg egy pont. Ez lesz az úrhajó. A kontroller bal és jobb gombjával lehessen balra és jobbra mozgatni. Fentről érkezzen egy meteor véletlenszerű helyről. A meteorral való ütközést el 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. A meteorok egyre gyorsabban érkezzenek, ahogy haladunk a játékban.

Sajnos a :GAME ZIP kijelzőn nem működik a játékfejlesztésnél hasznos sprite-os megoldás, így többet kell programoznunk.

Kezdetben inicializáljuk a kijelzőt, amelyhez a *neo* változót használjuk. Beállítjuk a meteor és az úrhajó koordinátáit. A késleltetés alap értéke legyen 500, ezt később csökkentjük, hogy nehezedjen a játékmenet. Kirajzoljuk a meteort pirossal, az úrhajót kékkel.

A pontszámot nullázzuk, és elindítjuk a visszaszámlálót 20 másodpercről. Megjelenítjük a változtatásokat a kijelzőn.



```
on start
  set neo to ZIP64 8x8 matrix display
  set meteorx to pick random 0 to 7
  set meteory to 0
  set urhajox to 3
  set urhajoy to 7
  set kesleltet to 500
  neo set matrix color at x meteorx y meteory to red
  neo set matrix color at x urhajox y urhajoy to blue
  set score 0
  start countdown (ms) 20000
  neo show
```

Beállítjuk, hogy Joypad balra gombjának megnyomásakor mi történjen. Csak akkor léphet balra az űrhajó, ha nem az 1. oszlopban áll (0-s indexű oszlop).

A régi helyéről letöröljük (feketével) az űrhajót, az új helyén megjelenítjük, kék színnel.

Hasonlóan készítjük el a Joypad jobb gombjához tartozó blokkokat, csak itt azt vizsgáljuk, hogy csak akkor léphet jobbra az űrhajó, ha nem az utolsó oszlopban áll.

```
on button Joypad Left (P12) press down
if urhajox > 0 then
  change urhajox by -1
  neo set matrix color at x urhajox + 1 y urhajoy to black
  neo set matrix color at x urhajox y urhajoy to blue
  neo show
```

```
on button Joypad Right (P13) press down
if urhajox < 7 then
  change urhajox by 1
  neo set matrix color at x urhajox - 1 y urhajoy to black
  neo set matrix color at x urhajox y urhajoy to blue
  neo show
```

A **forever** blokkba írjuk, hogy mi történjen a játék során.

A meteort lefele mozgatjuk és vizsgáljuk, hogy a meteor ugyanazon a koordinátán van-e, mint az űrhajó. Ha igen, akkor ütközés történt, így véget ér a játék.

Ha nem történt ütközés, akkor a meteor régi helyét töröljük feketével, és megjelenítjük az új helyen.

A megadott késleltetési értékkel lassítjuk a játékot.

Csökkentjük a késleltetés értékét, így a következő kör már gyorsabb lesz.

Ha a meteor az alsó sorba ért ütközés nélkül, akkor eltüntetjük, és egy véletlenszerű helyre rakjuk az első sorban.

Növeljük a játékos pontszámát.

```
forever
  if < meteory < 7 > then
    change meteory by 1
  +
  if << meteorx = urhajox >> and << meteory = urhajoy >> >> then
    neo set brightness 10
    neo show color red
    game over
    neo clear
  +
  neo set matrix color at x meteorx y meteory - 1 to black
  neo set matrix color at x meteorx y meteory to red
  neo show
  pause (ms) kesleltet
  if < kesleltet > 100 > then
    change kesleltet by -25
  +
  if < meteory = 7 > then
    neo set matrix color at x meteorx y meteory to black
    set meteorx to pick random 0 to 7
    set meteory to 0
    neo set matrix color at x meteorx y meteory to red
    neo show
    change score by 1
  +
```

Űrhajós játék - továbbfejlesztés

Feladat a diákok számára

Ötleteljetez arról, hogy még milyen továbbfejlesztési lehetőségek lehetnek ebben a játékban, és az általatok választottakat valósítsátok meg a hátralévő időben (egyénilag, vagy akár párokban is dolgozhattok).

Próbáljátok ki a társaitok által továbbfejlesztett játékokat!

Tanárként is javasolhatunk továbbfejlesztési ötleteket, amennyiben szükséges:

- Lehessen bekapcsolni az űrhajónak pajzsot. Ez jelentse azt, hogy pl. 2 ütközést kibír az űrhajó, és csak az utána lévő ütközéskor ér véget a játék.
- Legyenek hangeffektetek a játék során.
- Az űrhajó ne egy, hanem 3 egység széles legyen.
- Ütközéskor a rezgőmotor legyen elindítva.
- Ahogy nehezedik a játék, ne mindig véletlenszerű helyet válasszunk a meteornak, hanem az űrhajó x koordinátája legyen beállítva, hogy muszáj legyen arrébb lépni
- Több meteor érkezzen.
- Két játékos legyen és tároljuk el a két játékos pontszámát.

4. alkalom (Kapj el!)

| Tematikai egység | Alkalmazott munkaformák | módszerek, | Időtartam (perc) |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------------|
| 1. Kapj el! alapjáték elkészítése | Közös alkalmazásfejlesztés, magyarázattal | | 20 perc |
| 2. Az alapjáték továbbfejlesztése | Ötletelés, egyéni illetve páros munka | | 55 perc |
| 3. Az elkészült munkák megtekintése | A diákok bemutatják egymásnak az elkészült játékokat, amelyeket ki is lehet próbálni. Közös megbeszéljük az egyes munkák pozitívumait, és a lehetséges továbbfejlesztési ötleteket. | | 15 perc |



Kapj el!

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_Mh66wyhrfhbU

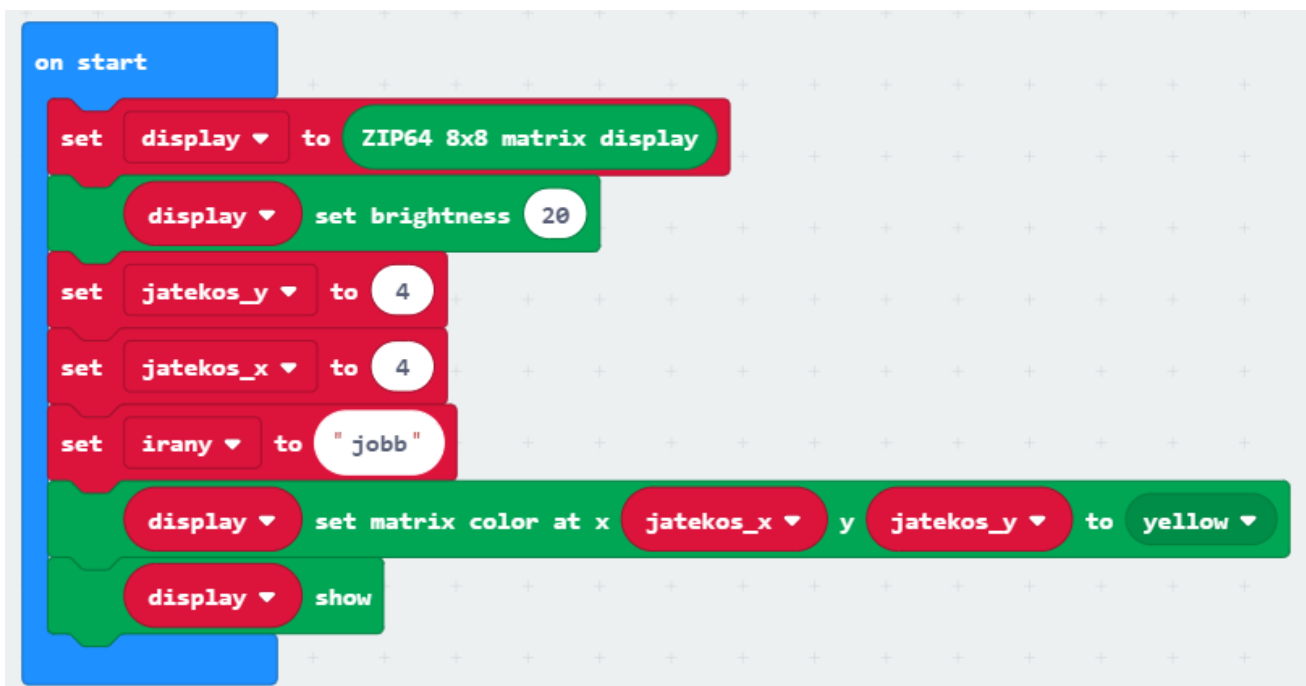
Nagyon sok játék épül arra, hogy egy adott játékosot kell irányítanunk a kijelzően úgy hogy az folyamatosan mozog, és a Joypad 4 gombjával tudjuk az irányát (fel, le, bal, jobb) megváltoztatni.

Most egy olyan alkalmazást készítünk közösen, amely ezt az alapjátékot valósítja meg, mégpedig úgy, hogy a falhoz érve a játékos iránya az ellenkezőjére változik.

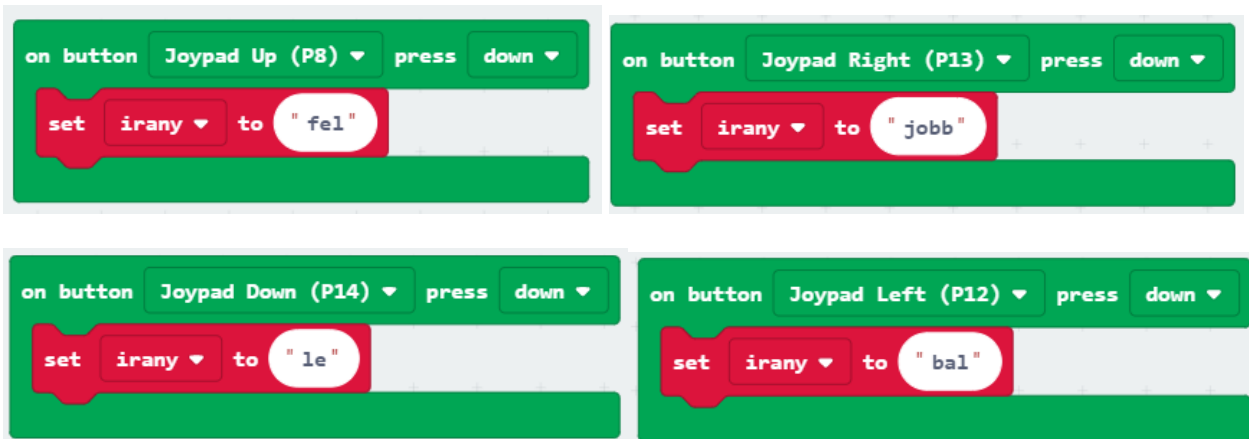
Az **on start** blokkban inicializáljuk a képernyőt, és beállítjuk a világosságértéket úgy, hogy ne vakítson a megjelenő pont. A 20-as érték megfelelő lehet.

Utána meghatározzuk a játékos kezdő x, illetve y koordinátáját, és beállítjuk az alapértelmezett irányt, ami jelen esetben a „jobb” lesz.

Ezek után megjelenítjük a sárga színű pontot az adott koordinátán, és frissítjük a kijelzőt (**show**).



Folytassuk azzal, hogy a 4 gomb lenyomására beállítjuk, hogy mely irányoknak feleljenek meg.

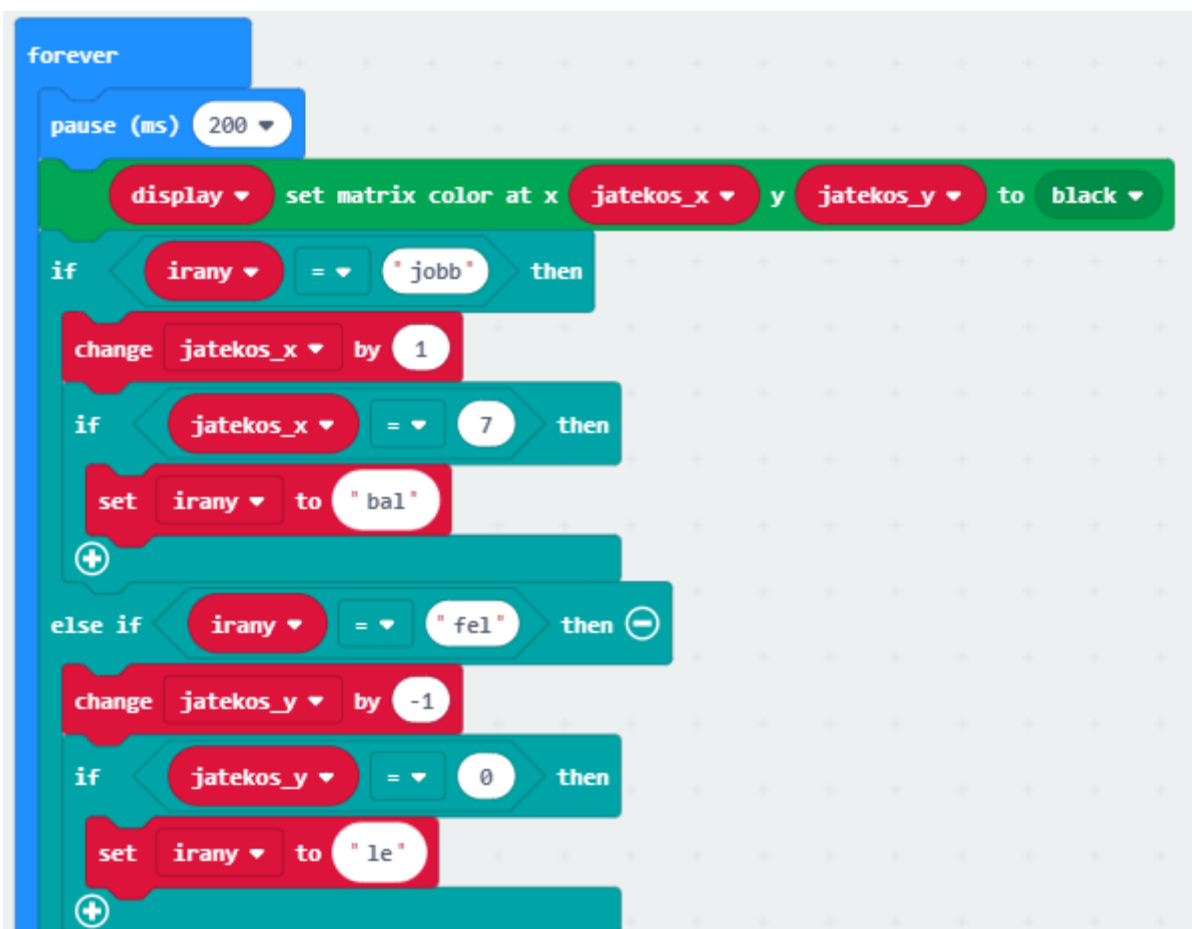


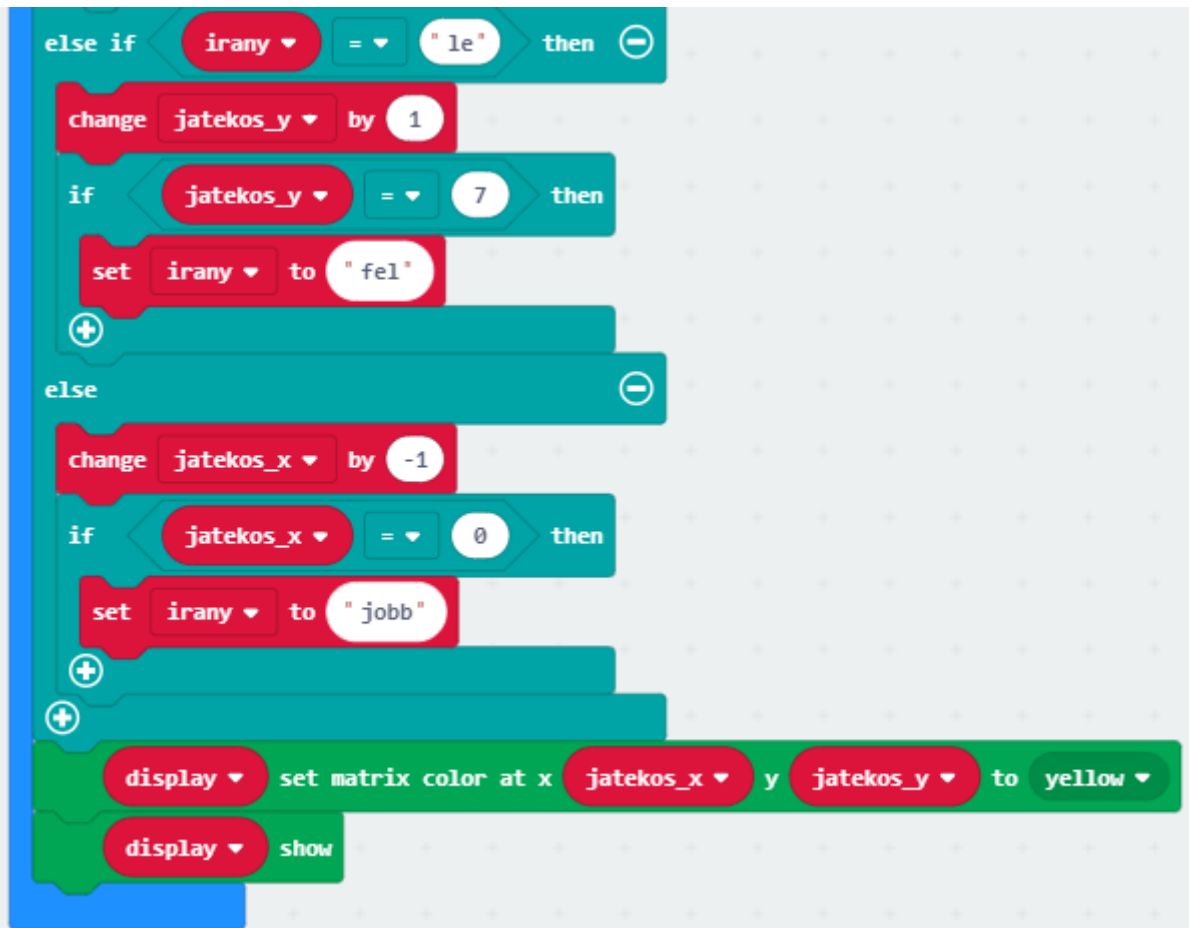
Ezt követően a **forever** blokkban megalkotjuk a játék működtetéséért felelős kódot.

Hogy ne legyen túl gyors a játékos mozgása, kismértékű várakozást állítunk be (200 ms). Ezek után az eredeti helyéről kitöröljük a pontot (feketével rajzoljuk ki).

Egy elágazással gondoskodnunk kell arról, hogy a megadott irányoknak megfelelően változzanak a pont koordinátái, illetve ha a pont eléri a kijelző szélét, akkor az irány automatikusan változzon meg az ellenkezőjére.

Az elágazás után kirajzoljuk az új helyre a pontot, és frissítjük a kijelzőt.





Alap játék továbbfejlesztése

Feladat a diákok számára

Ötleteljetez arról, hogy ezt az alapjátékot hogyan lehetne befejezni. Mi legyen a játék célja, mikor érjen véget a játék, hogyan számítjátok a pontokat, stb.

Valóítsátok meg ötleteiteket a hátralévő időben (egyénilég, vagy akár párokban is dolgozhattok).

Próbáljátok ki a társaitok által továbbfejlesztett játékokat!

Tanárként is javasolhatunk továbbfejlesztési ötleteket, amennyiben szükséges:

- Jelenjenek meg olyan színes pontok, amelyeket össze kell gyűjteni. Minden összegyűjtött elem növelje a játékos pontszámát. Nehezítés lehet, hogy bizonyos színű pontokat el kell kerülni, azok begyűjtésével véget ér a játék.
- Legyen egy másik játékos is, amely véletlenszerűen változtatja az útját. A cél, hogy elkerüljük az ütközést.

5. alkalom (Kígyó)

| Tematikai egység | Alkalmazott munkaformák | módszerek, | Időtartam (perc) |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------------|
| 1. Kígyó alapjáték elkészítése | Közös alkalmazásfejlesztés, magyarázattal | | 40 perc |
| 2. Az alapjáték továbbfejlesztése | Ötletelés, egyéni illetve páros munka | | 35 perc |
| 3. Az elkészült munkák megtekintése | A diákok bemutatják egymásnak az elkészült játékokat, amelyeket ki is lehet próbálni. Közös megbeszéljük az egyes munkák pozitívumait, és a lehetséges továbbfejlesztési ötleteket. | | 15 perc |



Kígyó - alapjáték

A tanár által elkészítendő/bemutatandó mintaprojekt

https://makecode.microbit.org/_96T7e17dyekY

A konzol játékok között nagyon népszerű a kígyó játék. Készítsük el közösen ezen játék alap változatát. Ez az alap hasonló lesz, mint az előző foglalkozáson látott (Kapj el) esetén, de most nem töröljük le azokat a pontokat, ahol már jártunk, illetve el is kell tárolnunk azon pontok azonosítóját, ahol már jártunk, hiszen a kígyó játék lényege, hogy nem ütközhetünk saját magunkba, így minden lépésnél vizsgálnunk kell, hogy üres helyre lépünk-e?

Hogy ne kelljen koordinátákat eltárolni, inkább hivatkozzunk az egyes pozíciókra a LED-ek sorszámával. A bal felső a nullás, a jobb alsó a 63-as. (ezek a sorszámok apró betűkkel rá is vannak nyomtatva a konzolra, a LED-ek fölé).

A diákok ötletei alapján beszéljük át, hogy mi a módja annak, hogy egy koordinátát átszámíthassuk sorszámmá, illetve hogy a sorszámból hogyan kaphatjuk meg a koordinátákat.

Az ötletelés során használhatjuk az alábbi táblázatot:

| X koordináta | Y koordináta | Sorszám |
|--------------|--------------|---------|
| 7 | 0 | 7 |
| 7 | 1 | 15 |
| 0 | 7 | 56 |
| 7 | 7 | 63 |

A megoldás:

Koordinátából sorszám számolása: $sorszám = y * 8 + x$

Sorszámból koordináták számolása:

$x = sorszám \bmod 8$ (8-al osztás után vett maradék)

$y = sorszám \div 8$ (8-al vett egészhányados)

A játék megvalósítása során az egyes sorszámokat egy tömbben helyezzük el. Úgy tudjuk majd eldönteni, hogy jártunk-e már az adott helyen, hogy lekérdezzük a tömbben az adott sorszám indexét. Ha az adott elem benne van a tömbben, akkor nullát, vagy nagyobb értéket kapunk (mivel a tömb nullától indexelődik), ha nincs benne, akkor pedig -1-et.

Lássuk a mintamegoldás magyarázatát:

```

on start
  set holjart to empty array
  set display to ZIP64 8x8 matrix display
  display set brightness 10
  set jatekos_y to 4
  set jatekos_x to 4
  set sorszam to (jatekos_y x 8 + jatekos_x)
  holjart add value sorszam to end
  set irany to "jobb"
  display set matrix color at x jatekos_x y jatekos_y to yellow
  display show

```

Az `on start` blokkban létrehozunk egy üres tömböt, *holjart* néven.

Inicializáljuk a kijelzőt, majd beállítjuk a világosságértéket. A kígyó kiindulási pontja a 4,4 koordináta lesz. Ezen koordinátát át kell számítanunk sorszámmá (*sorszam* változó).

Ezt a sorszámot elhelyezzük a *holjart* tömb végén.

Beállítjuk az alapértelmezett irányt jobbra, majd kirajzoljuk a sárga pontot és frissítjük a kijelzőt.

A kígyó irányváltoztatásáért ugyanazok a blokkok felelnek, mint az előző alapjáték esetén:

```

on button Joypad Up (P8) press down
  set irany to "fel"

on button Joypad Right (P13) press down
  set irany to "jobb"

on button Joypad Down (P14) press down
  set irany to "le"

on button Joypad Left (P12) press down
  set irany to "bal"

```

Amennyiben már jártunk egy adott sorszámú helyen, akkor véget kell érjen a játék, hiszen a kígyó nem ütközhet önmagába. Ezt a vizsgálatot több blokkban is el kell végeznünk, ezért érdemes ezt függvényként megvalósítani.

```
function jartmaritt? x y
  set sorszam to y * 8 + x
  if holjart find index of sorszam >= 0 then
    game over
```

Először is átszámítjuk a koordinátákat sorszámmá. Ha a tömbben már van ilyen elem, az azt jelenti, hogy már jártunk ott. Ez esetben vége a játéknak. Ezt legegyszerűbben úgy állíthatjuk be, ha a **Game** kategóriából a **game over** blokkot használjuk.

A **forever** blokk tartalma hasonló lesz az előző alapjátékhoz.

```
forever
  pause (ms) 200
  if irány = "jobb" then
    if jatekos_x < 7 then
      change jatekos_x by 1
      call jartmaritt? jatekos_x jatekos_y
    else
      game over
  else if irány = "fel" then
    if jatekos_y > 0 then
      change jatekos_y by -1
      call jartmaritt? jatekos_x jatekos_y
    else
      game over
  else if irány = "le" then
    if jatekos_y < 7 then
      change jatekos_y by 1
      call jartmaritt? jatekos_x jatekos_y
    else
      game over
  else if jatekos_x > 0 then
    change jatekos_x by -1
    call jartmaritt? jatekos_x jatekos_y
  else
    game over
  display set matrix color at x jatekos_x y jatekos_y to yellow
  display show
```

Egy kis várakozással indítunk (200 ms), majd az egyes irányoknak megfelelő vizsgálatot végzünk. Ha az adott irányban a kígyó elérte a szélét a játéktérnek, akkor vége a játéknak. Illetve azt is vizsgálunk kell, hogy nem lépett-e olyan helyre, ahol már járt korábban. Ha igen, akkor szintén vége a játéknak.

Az eljárás végén kirajzoljuk az új helyre a pontot és frissítjük a kijelzőt.

Kígyó alap játék továbbfejlesztése

Feladat a diákok számára

Ötleteljetez arról, hogy ezt az alapjátékot hogyan lehetne befejezni. Mi legyen a játék célja, mikor érjen véget a játék, hogyan számíttjátok a pontokat, stb.

Valósítsátok meg ötleteiteket a hátralévő időben (egyénilég, vagy akár párokban is dolgozhattok).

Próbáltjátok ki a társaitok által továbbfejlesztett játékokat!

Tanárként is javasolhatunk továbbfejlesztési ötleteket, amennyiben szükséges:

- A kapott pontszám legyen az, hogy mekkora területet sikerült lefedni. Ehhez lekérdezhettjük a *holjart* tömb hosszát (**length of array**).
- Megjelenhetnek speciális pontok a kijelzőn, amelyeket érintve nagyobb pontszámot kapunk.
- Megjelenhet olyan pont a kijelzőn, amelyet nem szabad érinteni, mert akkor véget ér a játék.

6. alkalom (Torpedó)

| Tematikai egység | Alkalmazott munkaformák | módszerek, | Időtartam (perc) |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|------------------|
| 1. Torpedó alapjáték elkészítése | Közös alkalmazásfejlesztés, magyarázattal | | 45 perc |
| 2. Az alapjáték továbbfejlesztése | Ötletelés, egyéni illetve páros munka | | 30 perc |
| 3. Az elkészült munkák megtekintése | A diákok bemutatják egymásnak az elkészült játékokat, amelyeket ki is lehet próbálni. Közös megbeszéljük az egyes munkák pozitívumait, és a lehetséges továbbfejlesztési ötleteket. | | 15 perc |



Torpedó - alapjáték

A tanár által elkészítendő/bemutatandó mintaprojekt

<https://makecode.microbit.org/i44McpPtieDA>

A torpedó játék is igen népszerű a konzol játékokon. Készítsünk el egy alapjátékot, ahol egy előre megadott pályán tudunk játszani. A torpedót (kék színű pont) a joystick gombjaival (fel, le, jobbra, balra) tudjuk irányítani. A Tűz 1 gombbal lövünk. Ha nem találtuk el a hajót, akkor zöld színű pont jelenik meg, ha eltaláltuk, akkor piros. A cél, hogy minél kevesebb lövéssel találjuk el az összes hajót.

A játék azzal indul, hogy rövid időre felvillan az összes hajó helyzete, amelyet el kell süllyeszteni. A pálya feltöltését úgy végezzük el, hogy a hajók nem érintkezhetnek egymással, még a sarkoknál sem. A pálya a következő darabszámú és hosszúságú hajókat tartalmazza:

- 1 db 5 hosszú hajó
- 2 db 4 hosszú
- 2 db 3 hosszút
- 2 db 2 hosszút
- 2 db 1 hosszút.

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |

Például az itt látható pálya megfelel a fenti leosztásnak:

A hajók helyzetét most úgy adjuk meg, hogy a LED-ek sorszámát felsoroljuk egy tömbben.

Például a fenti pálya esetén a tömb elemei a következők lesznek:

3,4,5,6,9,17,19,21,22,23,25,33,35,37,39,41,43,45,47,51,55,56,57,61,63

A játékot úgy célszerű megcsinálni, hogy minden lépésben kirajzoljuk, hogy milyen helyekre tippeltünk már, illetve milyen hajókat találtunk már el, és jelenleg melyik pozícióban tartózkodunk. Ezért a tippjeinket és a találatainkat is el fogjuk tárolni egy-egy tömbben.

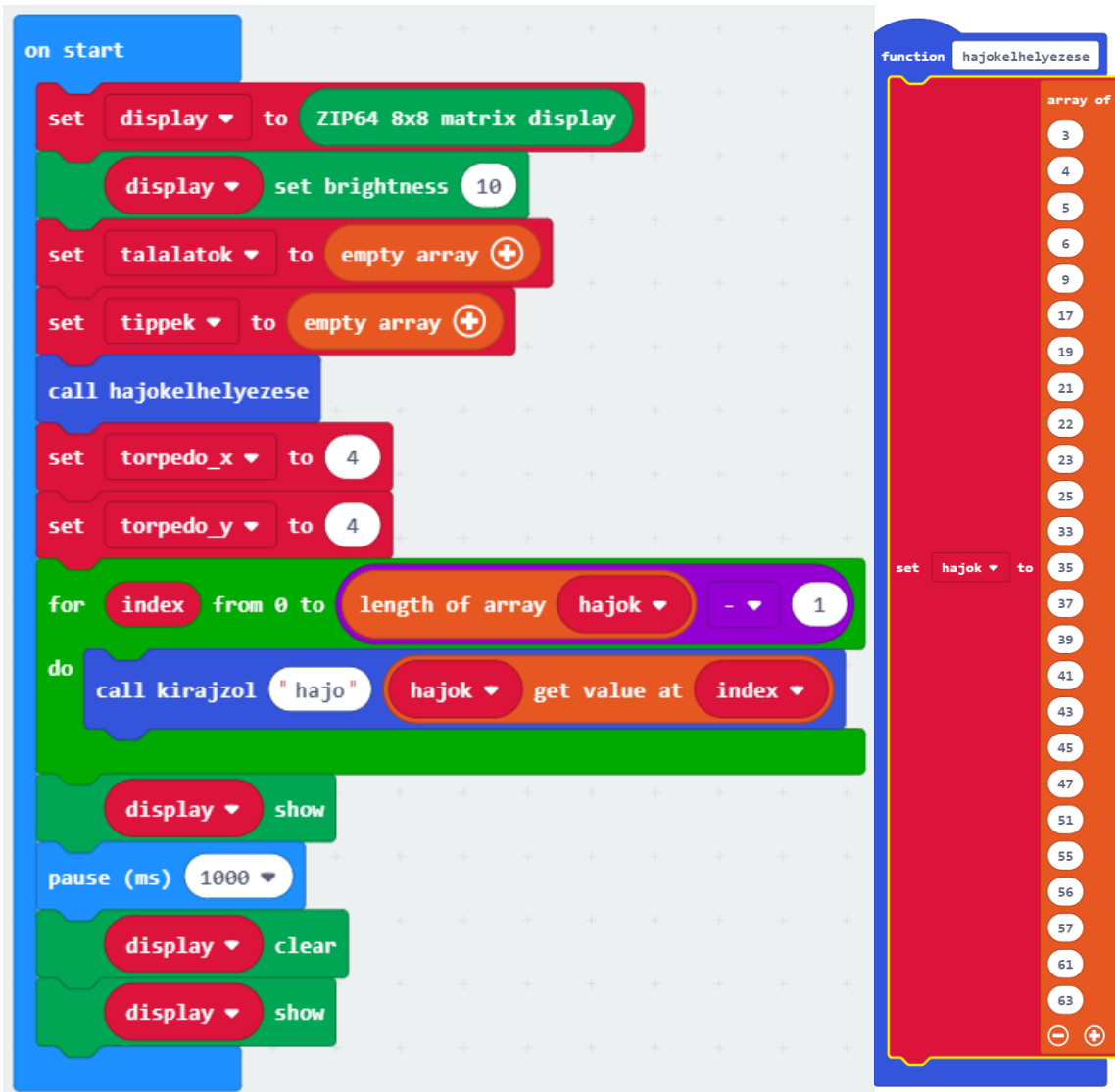
Ebben a példában a LED sorszámból ki kell számolnunk azt, hogy melyik koordinátára vonatkoznak, vagyis a korábban bemutatott számítást is fel kell használnunk:

Sorszámból koordináták számolása:

$x = \text{sorszám} \bmod 8$ (8-al osztás után vett maradék)

$y = \text{sorszám} \div 8$ (8-al vett egészhányados. Az osztás művelet egyébként így működik a Makecode környezetben.)

Lássuk a mintamegoldás magyarázatát:



Az `on start` blokkban inicializáljuk a kijelzőt, és beállítjuk a világosságértéket 10-re.

Szintén létrehozuk a találatok és a tippek nyilvántartására szolgáló tömböket.

A „*hajokelhelyezese*” nevű függvény felelős azért, hogy a megfelelő sorszámokat elhelyezze a hajok tömbben. Ezt a függvényt ebben a blokkban hívjuk meg.

Utána beállítjuk a torpedó kiinduló helyzetét, amely a 4,4 koordináta lesz.

Utána végigmegyünk a hajók tömbjén, és minden adott sorszámú helyre kirajzolunk egy piros pontot. Mivel a tömb 0-tól indexelődik, a tömb hosszaként pedig azt kapjuk vissza, hogy hány elemet tartalmaz a tömb, ezért a ciklusban a tömb hossza mínusz egy értékig megyünk. Mivel többféle színű pontot is rajzolnunk kell, attól függően, hogy hajóról, tippről, vagy találatról van szó, érdemes a kirajzolást is külön függvénybe szervezni. A függvénynek két paramétere lesz, egyrészt megadjuk hogy mit rajzolunk ki (hajó, tipp, találat), és hogy azt milyen sorszámú helyre kell kirajzolni. Ezen függvény magyarázatára később még kitérünk.

A rajzolás eredményét frissítenünk kell, ezért helyezzük el a `show` blokkot.

Az 1 másodperces késleltetés után eltüntetjük a kirajzolt hajókat (letöröljük a képernyőt - **clear**), majd a frissített képernyőt megjelenítjük (**show**).

Most nézzük a kirajzolásért felelős függvényt.

```
function kirajzol mit sorszam
set sorszambol_x to remainder of sorszam / 8
set sorszambol_y to sorszam / 8
if mit = "hajo" then
  display set matrix color at x sorszambol_x y sorszambol_y to red
else if mit = "tipp" then
  display set matrix color at x sorszambol_x y sorszambol_y to green
else
  display set matrix color at x sorszambol_x y sorszambol_y to red
```

A paraméterként megadott sorszámból kiszámítjuk a koordinátákat (*sorszambol_x*, *sorszambol_y*), és attól függően, hogy mit kell kirajzolnunk, más-más színeket használunk. A hajó pirossal, a tipp zölddel, a találat szintén pirossal lesz kirajzolva. (A találat van az elágazás utolsó ágában, hiszen ha valami nem hajó és nem tipp, akkor csak találat lehet).

```
on button Joypad Down (P14) press down
if torpedo_y < 7 then
  change torpedo_y by 1
```

```
on button Joypad Right (P13) press down
if torpedo_x < 7 then
  change torpedo_x by 1
```

```

on button Joypad Up (P8) press down
if torpedo_y > 0 then
change torpedo_y by -1

```

```

on button Joypad Left (P12) press down
if torpedo_x > 0 then
change torpedo_x by -1

```

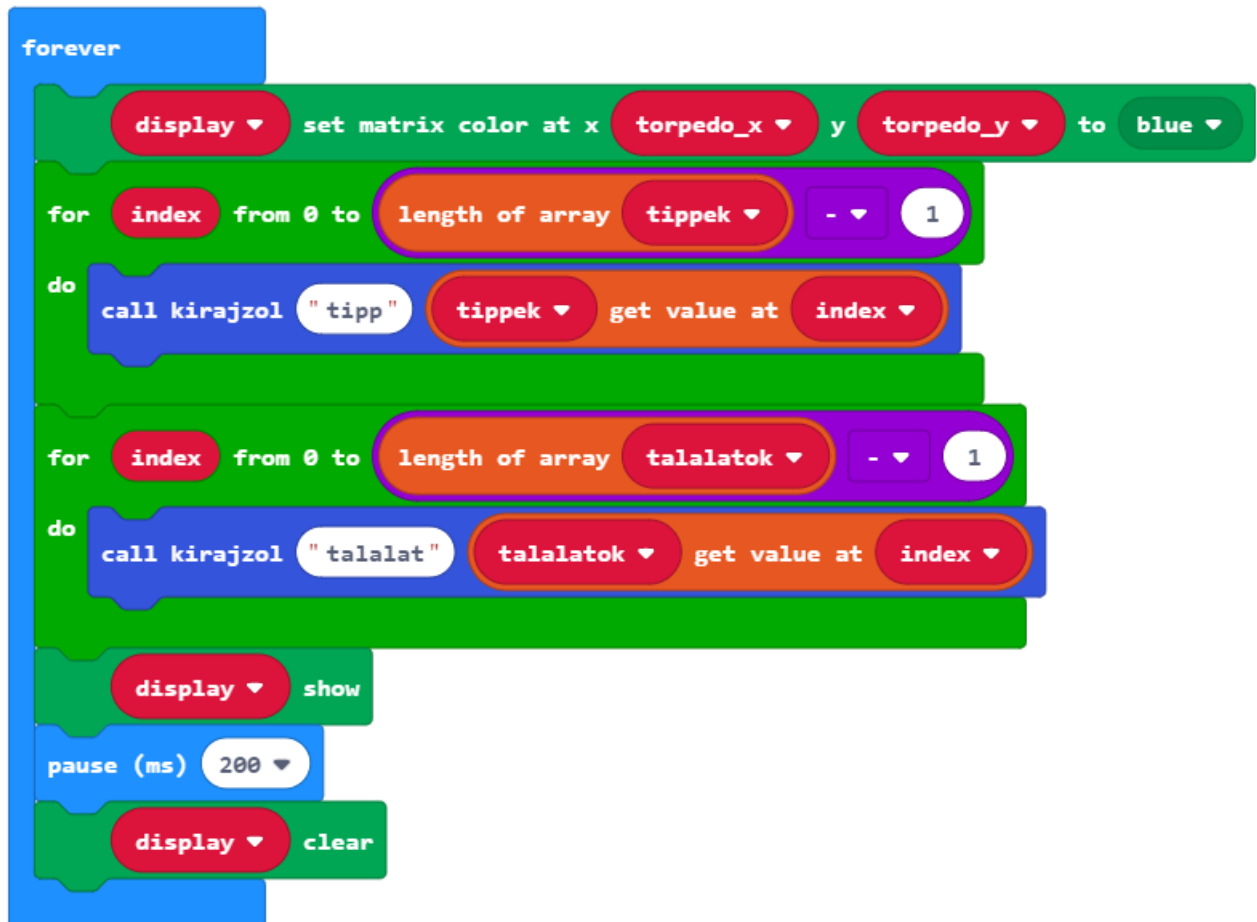
A Joypad (fel, le, balra, jobbra) billentyűinek kódját a már megszokott módon készítjük el. A pályáról nem mehetünk ki, ezért csak akkor növeljük, illetve csökkentjük az értékeket, ha még a pályán belülre kerülünk.

```

on button Fire 1 (P15) press down
set sorszam to torpedo_y * 8 + torpedo_x
if hajok find index of sorszam > 0 then
talalatok add value sorszam to end
else
tippek add value sorszam to end

```

A Tűz 1 gomb megnyomásáért felelős blokk kódjában azt látjuk, hogy a torpedó koordinátája alapján kiszámítjuk a LED sorszámot. Ha ezt a sorszámot megtaláljuk a hajók között, akkor sikerült eltalálni egy hajót, és betesszük a sorszámot a találatokat tartalmazó tömbbe. Ha ez nem áll fenn, akkor a tippeket tartalmazó tömbbe tesszük a sorszámot.



A játékot működtető logika a **forever** blokkba kerül.

Itt kirajzoljuk a torpedó helyzetét kék színnel, majd kirajzoljuk a már megadott tippek és találatok helyét, és frissítjük a kijelzőt. Kis késleltetés után pedig letöröljük azt.

Torpedó alap játék továbbfejlesztése

Feladat a diákok számára

Ötleteljetez arról, hogy ezt az alapjátékot hogyan lehetne befejezni. Mikor érjen véget a játék, hogyan számítjátok a pontokat, stb.

Valósítátok meg ötleteiteket a hátralévő időben (egyénilég, vagy akár párokban is dolgozhattok).

Próbáljátok ki a társaitok által továbbfejlesztett játékokat!

Tanárként is javasolhatunk továbbfejlesztési ötleteket, amennyiben szükséges:

- A micro:bit kijelzőjén jelenjen meg, hogy hány találatot értünk el.
- Ha eltaláltuk az összes hajót, legyen vége a játéknak és jelenjen meg egy animáció a micro:bit kijelzőjén.
- A pálya véletlenszerűen kerüljön meghatározásra.

Torpedó alap játék továbbfejlesztése (haladóknak)

Feladat a diákok számára

Alakítsátok át úgy az alkalmazást, hogy két micro:biten egymás ellen lehessen játszani. Az egyik micro:biten lehessen beállítani a pályát, a másikon pedig lehessen játszani.

További alkalmak

Ha több alkalomból álló szakkör tartására is van lehetőségünk, akkor bátran adjunk teret az önálló játékörök megvalósítására. Az ötletelés történhet párokban. Minden pár mutassa be az elgondolásait. Ügyeljünk arra, hogy a kigondolt feladat ne legyen túlságosan komplex, ilyen esetben javasoljunk egyszerűsítéseket a játékkal kapcsolatban.

Néhány ötlet

- Készítsünk egyszámjátékot. A micro:bitek kommunikáljanak egymással rádiókapcsolaton keresztül. Minden micro:bit küldjön át egy számot 0 és 63 között az összesítést végző micro:bitnek, amely :GAME ZIP controllerhez van kapcsolva. A kijelzőn zöld színnel jelenjen meg az adott sorszámú LED, ha arra csak 1 ember gondolt, és pirossal azok, amelyekre többen is gondoltak. Az nyer, aki a legkisebb olyan számra gondolt, amelyre senki más nem.
- Jelenítsünk meg animációkat a :GAME ZIP kijelzőjén. Találjuk ki, hogy mi lenne a legjobb módja annak, hogy az egyes animációs fázisokat (pontok koordinátája és színe) eltárolhassuk!
- Készítsünk reakciójátékot! Ha a kijelző felső részén jelenik meg pont, akkor a fel gombot kelljen megnyomni, ha jobb oldalán, akkor a jobb gombot, és így tovább. Minden helyes gombnyomásakor növeljük a játékos pontszámát, és minden rossznál csökkentjük.
- Készítsünk hangulatvilágítást! A rádiókapcsolaton kommunikáló micro:bitek küldhessenek egy színkódot és egy világosságértéket a :Game ZIP controllernek, és a kijelzőn az a szín jelenjen meg a megadott világosságértékkel. Fejlesszük tovább úgy az alkalmazást, hogy ne csak színt lehessen átküldeni, hanem azt is, hogy milyen sorszámú dallam játszódjon le a beépített dallamok közül.