1. alkalom (információ megjelenítése a LED mátrixon, animációk)

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam
Az eszköz bemutatása, munkavédelmi előírások	Frontális tanári bemutató	8 perc
A szerkesztőfelület bemutatása	Frontális tanári bemutató, a szoftver legfontosabb jellemzőinek megismerése	6 perc
Hello, World! program	Közös programírás, tanári magyarázattal	8 perc
Képek megjelenítése	Tanári bemutató, majd egyéni kísérletezés Egyéni képek megjelenítése	5 perc
Animációk megjelenítése	Tanári magyarázat a listáról, animáció beépített listákkal, animáció saját képekkel Egyéni, kreatív feladat, tanári segítséggel, a munkák bemutatása egymásnak	30 perc
Gombok kezelése	Tanári bemutató és magyarázat	30 perc

1. Munkavédelmi előírások, az eszköz bemutatása

Az eszközök kiosztása előtt fontos felhívni a diákok figyelmét pár szabályra, főleg, ha még nem dolgoztak hasonló eszközzel. A munkavédelmi előírások megtalálhatóak a Programozzunk micro:biteket! kiadvány 21. oldalán [5].

Ezek után mutassuk be az eszközt a diákoknak, kitérve a rajta található gombokra, csatlakozókra, érzékelőkre. A számítógéphez való csatlakozás Bluetooth-on keresztül, vagy USB kábel segítségével történhet meg. Az utóbbi mód egyszerűbb, hiszen ilyenkor az áramellátás is megoldódik, nem kell elemet csatlakoztatnunk az eszközhöz. A csatlakozás módja hasonló pl. egy okostelefonéhoz, így a diákok számára biztosan ismerős lesz. Hívjuk fel a figyelmet arra, hogy az eszköz egy külön meghajtóként jelenik

meg, akárcsak egy pendrive. Erre a meghajtóra programokat másolva tudjuk azokat futtatni a micro:biten.

2. A szerkesztőfelület bemutatása

Az eszköz megismerése után írjuk meg közösen az első programunkat! Ehhez először szükség van a szerkesztőfelület megismerésére. Python alapon is több lehetőségünk van a micro:bit programozására. Egyik lehetőségként választhatjuk a böngészőben használható szerkesztőt[4]. Itt azonnal, telepítés nélkül neki lehet állni kódolni, ami bizonyos körülmények között jó választásnak bizonyulhat. Azonban hamar hiányérzetünk támadhat, mivel a felület jelenleg sem automatikus kódkiegészítést, sem hibakeresési lehetőségeket nem támogat. Az esetleg szintaktikus hibáinkat is csak a micro:bit kijelzőjén futó szövegből hámozhatjuk ki, ami jelentősen lelassítja a munkát. Ezen okok miatt inkább egy asztali szerkesztő javasolt. A támogatott választás a Mu névre hallgató program[5], ami beépített micro:bit üzemmóddal segít nekünk. Ez automatikus kódkiegészítést és magyarázatot tartalmaz, emellett képes a kódot egy kattintással a micro:bitre másolni, valamint ellenőrizni tudja a kódunk helyességét is. Haladó lehetőségei közül érdemes lehet megismerni még a REPL-t és a plotter-t is, ám ezekkel ebben az anyagban nem foglalkozunk.

A diákoknak röviden mutassuk be a program legfontosabb funkciót: mód választása, új program létrehozása, a mentés és a flash-elés közti különbség, valamint a kód hibaellenőrzése gomb. Ezek után készen is állunk az első programunk megírására!

3. Hello, World! program

Ha már programoztak a diákok, biztosan ismerős számukra a *Hello, World!* kifejezés. Első programunk ezt a szöveget fogja kiírni a micro:bit LED mátrixán. Gépeljük be közösen az alábbi kódot:

```
from microbit import *
display.scroll('Hello, World!')
```

Magyarázzuk el a diákoknak, hogy minden programot az úgynevezett importálással kell kezdenünk, itt mondjuk meg a programunknak, hogy a micro:bit kódkönyvtárát szeretnénk használni. *(Megjegyzés: az importálás az anyagban található többi kódból*

[4] https://python.microbit.org Elérés dátuma: 2018.08.03.
[5] https://codewith.mu/ Elérés dátuma: 2018.08.03.

helytakarékossági okok miatt kihagyásra kerül, azonban ezt minden programunknál használjuk!) A szöveg kiírásához a `display` osztály `scroll()` metódusát használtuk, aminek az első, és jelen esetben egyetlen paraméterében a kiírandó szöveget kell megadnunk.

Egészítsük ki a programunkat úgy, hogy a szöveget ne csak egyszer írja ki a micro:bit, hanem újra és újra, két másodperces szüneteket tartva az egyes kiírások között. Ehhez egy végtelen ciklusba kell tennünk a kiírást, valamint a `sleep()` függvényt használva szüneteltetni a programot. A kód:

```
while True:
    display.scroll('Hello, World!')
    sleep(2000)
```

Nézzük meg a ciklus szintaktikáját a Pythonban! A `while` kulcsszóval kell kezdenünk, ezután jön a bennmaradási feltételünk, majd kettősponttal jelezzük, hogy a ciklusmag következik. A ciklusmag utasításai új sorba, beljebb kezdve kell írnunk. Pythonban más népszerű programozási nyelvekkel ellentétben nem kapcsos zárójelek közé kell tennünk egy-egy ilyen blokkot, hanem behúzásokkal kell jeleznünk az összetartozást. A szép kód írása ezáltal elkerülhetetlen.

4. Képek megjelenítése

Ezek után folytatjuk a megismerkedést a kijelző kezelésével, amihez továbbra is a `display` osztály függvényeit használjuk. A kijelző nem csak szövegek, hanem képek megjelenítésére is használható. A következő programmal jelenítsünk meg egy mosolygó arcot a LED-ek segítségével! Ehhez a `display.show()` metódust fogjuk használni, ami képek megjelenítésére is használható:

```
display.show(Image.HAPPY)
```

A függvény paraméterében a megjeleníteni kívánt képet adjuk meg. Látható, hogy a micro:bit modul beépített képeket is tartalmaz, amiket az `Image` osztály attribútumai segítségével tudunk használni. A teljes lista megtalálható a dokumentáció `Image` osztályt bemutató részében[6]. A listát mutassuk meg a diákoknak is, kísérletezzenek a különböző képekkel, nézzék meg, hogy melyik kép pontosan hogyan jelenik meg, mit ábrázol.

[6] https://microbit-micropython.readthedocs.io/en/latest/image.html Elérés dátuma: 2018.08.03.

Biztosak lehetünk benne, hogy a gyerekek kíváncsisága nem áll meg itt, saját maguk által alkotott képet szeretnének megjeleníteni a kijelzőn. Természetesen erre is van lehetőség. Minden egyes LED-hez tartozik egy 0 és 9 közötti érték, ami azt jelzi, hogy milyen fényerővel világít az adott LED. A 0 jelenti a kikapcsolt állapotot, a 9 a legfényesebbet. Ezek segítségével van lehetőségünk saját képek megjelenítésére, például kirajzolhatunk egy egyszerű repülőt, az alábbi módon:

```
plane = Image("00900:"
              "09990:"
              "90909:"
              "00900:"
              "09990")

display.show(plane)
```

Hívjuk fel a diákok figyelmét a helyes kódolásra; elsőre talán bonyolultnak tűnhet a zárójelek, idézőjelek, és a kettőspontok használata, főleg, ha még nem programoztak szövegesen.

Talán a legbeszédesebb mód a fenti az egyéni képek létrehozására, azonban miután rutinosabban lettünk ebben, használhatjuk az alábbi, rövidebb formát is:

```
plane = Image("00900:09990:90909:00900:09990")
```

Próbálkozhatunk a különböző fényerősségek hatásával, például átírhatunk minden számot 1-esre.

Feladat a diákok számára

Alkossatok párokat, majd mondjátok meg egymásnak, hogy mit rajzoljon ki a párotok a LED mátrixon, a fenti módszert használva!

5. Animációk megjelenítése

A micro:bit kijelzőjén animációkat – képek sorozatát – is nagyon egyszerűen megjeleníthetjük. Ehhez mindössze a képsorozatra van szükségünk, ami a Pythonban nagyon jól megadható az ún. listával. Ha a diákok még nem találkoztak a lista, vagy a tömb fogalmával, magyarázzuk el nekik! Hozzunk hétköznapi példákat, például egy bevásárlólista, egy dolgozat jegyeinek a listája stb. A lista a Pythonban azonban bármilyen típusú elemekből állhat, így képekből is. Próbáljunk meg egy ilyen listát animáció formájában megjeleníteni. Az eddig használt `Image` osztályban két beépített

lista is van, név szerint az `Image.ALL_CLOCKS`, és az `Image.ALL_ARROWS`. Csináljunk egy olyan programot, ami folyamatosan megjelenít egy járó órát!

```
display.show(Image.ALL_CLOCKS, loop=True, delay=100)
```

A korábban már megismert `show()` függvény egy új formáját használjuk. Az első paraméterben valamilyen iterálhatót kell megadnunk, aminek az egyes elemeit fogja a függvény megjeleníteni. A `loop` paramétert igazra állítva végtelenítheti jük az animációt, a `delay` paraméter pedig az egyes állapotok közötti időt adja meg, milliszekundumban. Hívjuk fel a figyelmet rá, hogy a `loop` és a `delay` paramétereknél nem elég csak az értékeket megadni, a nevüket is fel kell tüntetni! Ennek oka, hogy bizonyos paramétereknek alapértelmezetten is van értékük, ezért, ha valamelyiket át szeretnénk állítani, tudnia kell az értelmezőnek, hogy melyik paraméter értékét állítjuk át.

Feladat a diákok számára

Készítsetek animációt, ami a különböző nyilakat jeleníti meg a kijelzőn! Használjátok az `Image` osztály `ALL_ARROWS` listáját! Az animáció ne ismétlődjön, de legyen negyed olyan gyors, mint az órás! A végén töröljétek a kijelzőt a `clear` paraméter segítségével!

Ezután mutassuk meg a diákoknak, hogy hogyan tudnak létrehozni saját maguk listát, általuk megadott értékekkel, hogy aztán az elemeit felhasználják saját animáció készítéséhez. Példaként a korábban már kirajzolt repülő fog „elrepülni”:

```
plane1 = Image("00900:09990:90909:00900:09990")
plane2 = Image("09990:90909:00900:09990:00000")
plane3 = Image("90909:00900:09990:00000:00000")
plane4 = Image("00900:09990:00000:00000:00000")
plane5 = Image("09990:00000:00000:00000:00000")

plane_list = [plane1, plane2, plane3, plane4, plane5]

display.show(plane_list, delay=200, clear=True)
```

Feladat a diákok számára

Készítsetek egy rövid animációs „filmet” saját képekből, esetlegesen szöveggel kísérve! A képeket tároljátok egy listában! Használjátok ki a `display.show()` metódus különböző paramétereit! Ne feledkezzetek meg arról sem, hogy a fényerősség is állítható!

6. Gombok kezelése

A `display` osztály további függvényeinek megnézése előtt tanuljuk meg, hogy hogyan kezelhetjük a micro:bit előlapján található két gombot. Ezekkel gyakorlatilag az eseményvezérelt programozás alapjaiba vezethetjük be a diákokat, hiszen a gombok lenyomására fog lefutni egy adott kódrészlet. A másik fontos különbség az eddigi kódokhoz képest, hogy most már nem csak kimentet állít elő a program, hanem reagálnia kell valamilyen bemenetre is. Érdemes erre is felhívni a diákok figyelmét. Gépeljük be a következő, pár soros kódot, majd nézzük át a diákokkal, hogy miket figyelhetünk meg benne.

```
sleep(10000)
display.scroll(str(button_a.get_presses()))
```

Az első sort már a diákoknak is ismerniük kell: 10 másodpercig megállítja a program futtatását. A `display` osztály `scroll()` metódusát már szintén ismerik. Ennek a paraméterében találhatóak az új dolgok. Kezdjük legbelülről! A `button_a` osztály kezeli a micro:bit „A” jelű gombját. Ennek az osztálynak az egyik metódusa a `get_presses()`, ami visszaadja, hogy hányszor lett lenyomva az adott gomb a program futásának kezdete óta, emellett lenullázza ezt az értéket. Ugyanígy lehet a „B” gombot is kezelni, a `button_b` osztállyal. A blokkos szerkesztővel[7] ellentétben itt nincs külön opció a két gomb egyszerre történő kezelésére, ezt logikai operátorokkal kell megoldanunk. Erre egy későbbi feladat során még visszatérünk.

Ha megkaptuk, hogy hányszor lett lenyomva a gomb, már csak az a feladatunk, hogy kiírjuk ezt a kijelzőre. A `scroll()` metódus egy szöveget (stringet) vár paraméteréül, a `get_presses()` pedig egy számot ad vissza, így ezt át kell alakítanunk. Erre szolgál az `str()` függvény, aminek a paraméterében adhatjuk meg a konvertálni kívánt számot. Jelenleg ez a függvény akár el is hagyható, mivel, ha számot adunk meg, a `scroll()` metódus automatikusan konvertál, de a későbbiekben mindenképpen hasznos tudás lehet a típuskonverzió fogalmának bevezetése (vagy lehet, hogy már elő is fordult korábban). A metódusok, függvények egymásba ágyazása elsőre bonyolultnak tűnhet a diákok számára, legyünk biztosak benne, hogy megértették, hiszen ez egy nagyon gyakran használt lehetőség a programozás világában.

[7] https://makecode.microbit.org/ Elérés dátuma: 2018.08.03.

Pihenésképpen versenyeztessük meg a diákokat, nézzük, hogy ki tudja a legtöbbször lenyomni a gombot ebben a 10 másodpercben!

Feladat a diákok számára

Módosítsátok a az előző programot úgy, hogy a 10 másodperc letelte után az jelenik meg, hogy összesen hányszor nyomtuk le a két gombot! Ezután módosítsd úgy, hogy csak 5 másodpercünk legyen a gombok nyomkodására!