2. alkalom (a kijelző további lehetőségei, események kezelése ciklussal)

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam
Ismétlés	Egyéni munka, majd a megoldás megbeszélése közösen	10 perc
A `display` osztály további metódusai	Frontális tanári magyarázat, közös programírás, önálló munka, játék	50 perc
Események kezelése ciklussal	Közös programírás, tanári magyarázattal	30 perc

1. Ismétlés

Feladat a diákok számára

Készítsetek programot, ami minden ötödik másodpercben kiírja, hogy hányszor nyomtuk le a „B” gombot az előző öt másodperces szakasz alatt!

A megoldás:

```
while True:
    sleep(5000)
    display.scroll(button_b.get_presses())
```

2. A display osztály további metódusai

A második alkalmon folytatjuk az ismerkedést a `display` és az `Image` osztályokkal. Elsőként nézzük meg, hogy milyen további metódusokat kínál a `display` osztály a kijelző kezelésére!

Ennek az osztálynak a segítségével lehetőségünk van a kijelző egyes LED-jeinek külön-külön való kezelésére is. Ebben az esetben minden egyes LED-et koordinátákkal azonosíthatunk, a mátrix típushoz hasonló módon. A diákok a tömb fogalmát már ismerik, ennek alapján magyarázzuk el nekik, hogy mi az a mátrix, kitérve az x és y koordinátákra. A micro:bit kijelzőjén a bal felső sarokban található a 0,0 koordinátájú LED, a jobb alsó sarokban pedig a 4,4 koordinátájú. Ismétlésképpen hívjuk fel a figyelmüket arra is, hogy az indexelés 0-tól kezdődik! Példaként rajzoljunk fel egy 5*5-

ös táblázatot a táblára, ami a micro:bit kijelzőjét szimbolizálja, majd ezen mutassunk rá cellákra, és kérdezzük meg a diákokat, hogy az melyik koordinátájú LED lenne a micro:biten.

Ennek a tudásnak a birtokában máris használni tudjuk a `display` osztály `get_pixel()` és `set_pixel()` metódusait. Előbbi visszaadja a paraméterekben megadott indexű LED világosságértékét (emlékezzünk, hogy ez egy 0 és 9 közötti szám), utóbbival pedig be lehet állítani a paraméterekben megadott LED világosságértékét egy, szintén a paraméterekben megadott értékre. Használjuk most ezeket egy programban, ahol egy új vezérlési szerkezetet is láthatunk! Ha korábban már programoztak a diákok, biztosan szóba került az elágazás fogalma. Röviden ismételjük át ezt velük még a kód leírása előtt, majd gépeljük be az alábbi sorokat, ahol láthatjuk, hogy a Python nyelvben hogyan is kell használni az elágazást:

```
if display.get_pixel(1,1) == 0:
    display.set_pixel(1,1,9)
sleep(2000)
display.clear()
```

Mint láthatjuk, az elágazás itt is az `if` kulcsszóval kezdődik, ezután kell írnunk feltételt, zárójelet nem használva. A feltétel után kettősponttal jelezzük, hogy az elágazás törzse következik; ezt új sorba, beljebb kezdve kell írnunk. Vegyük észre, hogy a szintaktika teljesen megegyezik a már használt `while` ciklussal! Amennyibben nem programoztak még C típusú nyelvben a diákok, akkor érdemes felhívni a figyelmüket az egyenlőség vizsgálatára használt dupla egyenlőségjelre, szembe állítva az értékadáshoz használt szimpla verziójával. A behúzásokból láthatjuk, hogy a `sleep()` függvény már nem része az elágazásnak, az mindenképpen lefut, ahogy a kijelző törlésére szolgáló `clear()` metódus is.

Ezek után gyakoroljuk a LED mátrix kezelését. Gondoljunk most úgy a micro:bit kijelzőjére, mintha ablakok lennének! Készítsük el egy emeletes ház szimulációját, amin azt láthatjuk, ahogy az egyes lakásokban a fények kigyúlnak és elalszanak! Illusztrációként bemutathatunk egy videót a jelenségről[8]. Időtakarékossági szempontok miatt használjuk csak a kijelző első három oszlopát! Ezt az 5 pixel magas és 3 pixel széles

[8] https://www.youtube.com/watch?v=M-34aBULH8g Elérés dátuma: 2018.08.03.

területet fogjuk a háznak tekinteni. Az eddigi ismeretek alapján a feladat kiadható a diákoknak önálló megoldásra.

Feladat a diákok számára

A ház földszintjén üzletek találhatóak, ezek mindig világítsanak. Ezek után kapcsolj fel pár LED-et (szabadon választott koordinátákkal), majd kapcsolj is le közülük néhányat! Módosítsd úgy a programot, hogy először az első emelet lakásaiban kapcsoljanak fel a fények balról jobbra, majd a második emeleten jobbról balra! A lámpák lekapcsolása ugyanilyen sorrendben történjen meg!

A következő két metódus a kijelző ki és bekapcsolására szolgál, ezeket a beszédes `on()` és `off()` nevekkel látták el. Feldolgozásuk történhet bármilyen önálló feladat keretében, de használható példa lehet egy bombás játék megalkotása. A játék lényege, hogy a diákok körbe állnak, egymásnak adogatva a „bombát”, ami jelen esetben egy micro:bit lesz. A bombát akkor lehet továbbadni, hogyha mondtak egy fogalmat a megbeszélt témában. Az érettségi tételek gyakorlására kiváló játék lehet (pl. fogalmak a háttértárak témaköréből). Akinél „felrobban a bomba” (megjelenik a kijelzőn a bomba alakzat), felrobbant, és kiáll a játékból. Az utolsóként életben maradt játékos nyer. A játékhoz szükséges programot a diákok önálló feladat keretében is megcsinálhatják:

Feladat a diákok számára

A program elindulásakor rajzolódjon ki valamilyen bombához hasonló alakzat a kijelzőn, ez villanjon fel háromszor, majd kapcsoljátok ki a kijelzőt a `display.off()` metódussal. Csoportlétszámtól függően körülbelül fél-egy perc elteltével a kijelző kapcsoljon be (a kijelzőn a bomba lesz látható).

Játszható akármilyen témakörben, akár hétköznapi is lehet, akár angolul stb. A lehetőségek száma gyakorlatilag végtelen!

Egy másik példa lehet erre a két metódusra egy zsákbamacska készítése.

Feladat a diákok számára

A micro:bit kijelzőjére rajzoljatok valamilyen egyedi ikont, valami olyan témában, ami rátok jellemző. Az A gombra a kijelző kapcsoljon ki, a B gomb megnyomására pedig kapcsoljon be! Miután a program fut, kapcsoljátok ki a kijelzőt, majd minden tegye bele egy zsákba a micro:bitjét! Ezután a zsákból mindenki húz egy micro:bitet, bekapcsolja a kijelzőt, majd a rajta található ikon segítségével ki kell találnia, hogy kinek a micro:bitjét húzta ki.

A fenti feladat jól használható például párokba osztásnál: ilyenkor csak minden második diák micro:bitje kerül bele a zsákba, a diákok másik fele pedig kihúzza azokat. Akinek az eszközét húzta, az lesz a párja.

3. Események kezelése ciklussal

Gyakran van szükségünk arra, hogy a programunk várjon valamilyen esemény bekövetkeztére, például egy gombnyomásra. Ehhez ciklusba szervezünk egy kóddarabot, ami megmondja, hogyan kell reagálni a várt eseményekre. A korábban már használt `while` ciklust fogjuk erre a célra használni, ami a ciklusfeltétel teljesülése esetén futtatja a ciklusmagot. Ennek demonstrálására nézzünk egy egyszerű kódot:

```
while running_time() < 10000:
    display.show(Image.ASLEEP)

display.show(Image.SURPRISED)
```

A ciklus feltételében a `running_time()` függvénnyel kérdezzük le a program elindítása óta eltelt időt. Természetesen ez is milliszekundumban adja meg az eredményt. Ha még nem telt el 10 másodperc az elindítás óta, a micro:bit „alszik”. Amint letelik ez az idő, a micro:bit meglepődött arccal „felébred”. Ismét hívjuk fel a diákok figyelmét a szintaxisra, különös tekintettel a behúzásokra, hiszen a Python nyelvben csakis ezen múlik, hogy mely sorok alkotnak egy blokkot a kódon belül.

Ezek után áttérhetünk első eseményünkre, a gombnyomásra. Ha azt akarjuk, hogy a micro:bit reagáljon valamit a gombnyomásunkra, egy végtelen ciklus belül kell vizsgálnunk azt, hogy le van-e nyomva éppen a gomb. Erre szolgál a `Button` osztály `is_pressed()` metódusa. Készítsünk most egy kezdetleges virtuális állatot, ami folyamatosan szomorú, kivéve amikor az „A” gomb le van nyomva, ilyenkor vidám lesz.

Amennyiben a „B” gombot nyomjuk le, az állat elalszik (letörlődik a kijelző, és véget ér a program). Az ehhez szükséges kód:

```
while True:
    if button_a.is_pressed():
        display.show(Image.HAPPY)
    elif button_b.is_pressed():
        break
    else:
        display.show(Image.SAD)

display.clear()
```

Vizsgáljuk meg sorról sorra a programot! Az első sorban indítjuk a végtelen ciklust, ami „hallgatózik” az események (a gombok lenyomása) után. A ciklusmagban elágazással vizsgáljuk, hogy az „A” gomb le van-e nyomva éppen. Amennyiben igen, a kijelzőn egy boldog arcot jelenítünk meg. Az `elif` kulcsszó felel meg a más nyelvekben `else if`-ként megszokott `különben ha` ágnak. Ha az „A” gomb épp nincs lenyomva, de a „B” igen, lépjünk ki a ciklusból, és folytatódjon a program futása. Az éppen futó kódblokkból tehát – más nyelvekhez hasonlóan – a `break` kulcsszóval tudunk kiugrani. Ha egyik gomb sincs lenyomva, a szomorú arcot rajzoljuk ki a kijelzőre. Ezt tehetjük a különben ágba, hiszen ide valóban csak akkor jutunk, ha egyik gomb sincs lenyomva. A ciklus után letöröljük a kijelzőt (az állat elalszik), majd véget ér a program.

A második alkalom végén tartsunk egy kis összegzést: tudjuk, hogyan töltsünk rá programot a micro:bitre, milyen lehetőségeink vannak a kijelzőt illetően, hogyan használjuk a `while` ciklust, az elágazást, valamint a gombokat is tudjuk kezelni.

Ha esetleg marad idő a foglalkozás végén, a virtuális állat tetszés szerint továbbfejleszthető!