6. alkalom (iránytű, rádió)

Tematikai egység	Alkalmazott módszerek, munkaformák	Időtartam
Az iránytű	Frontális tanári magyarázat, közös programírás	20 perc
Rádió	Frontális tanári, magyarázat	5 perc
Szentjánosbogarak	Közös programírás	25 perc
Hideg-meleg játék	Közös megbeszélés, pármunka	40 perc

1. Az iránytű

Az eszközben található egy beépített iránytű, aminek segítségével meghatározhatjuk, hogy merre van észak. A diákokkal közösen írjuk meg az ezt megvalósító programot!

```
compass.calibrate()

while True:
    needle = ((15 - compass.heading()) // 30) % 12
    display.show(Image.ALL_CLOCKS[needle])
```

Mindig, amikor az iránytűt használjuk, kalibrálnunk kell az eszközt, hogy megfelelő értékeket kapjunk az érzékelőből. Erre szolgál a `compass` osztály `calibrate()` metódusa. Amikor lefut ez a metódus, a micro:bit megkér minket, hogy „rajzoljuk” tele a kijelzőt, az eszköz különböző irányokba való döntésének segítségével. Amennyiben ez sikerült, egy mosolygó fej a jutalmunk, és használatba vehetjük az iránytűt. Az állandó működést újból egy végtelen ciklussal biztosítjuk. Ezen belül egy kis matematika segítségével kiszámoljuk a `compass.heading()` metódus értékéből, hogy merre kell mutatnia a mutatónak. A `//` az egészosztás, a `%` a maradékos osztás jele a Pythonban. A `heading()` érték 0 és 360 fok között adja meg az aktuális irányt, ahol a 0 az észak. A ciklusmag második sorában a kiszámolt érték segítségével kirajzolunk egy órát, aminek a mutatója nagyjából mindig észak felé fog mutatni. A diákok összehasonlíthatják az egyes eszközök eredményeit, így ellenőrizve, hogy jól működnek-e, illetve, hogy jó programot írtak-e.

2. Rádió

Mindeddig csak egyetlen eszközt használtunk a programjainkhoz. Azonban a micro:bitek rendelkeznek beépített rádió-adóvevővel is, aminek a segítségével kommunikálni tudnak egymással. Ez rengeteg új lehetőséget nyit meg, legyen az akár két eszköz közötti információcsere, többszemélyes, több eszközön futtatható játékok stb.

A rádiózással kapcsolatosan felmerülő probléma, hogy nem lehet közvetlenül egy személynek küldeni a jeleket, azokat mindenki tudja olvasni, aki megfelelő vevővel rendelkezik. Ezért fontos, hogy valamilyen szinten el tudjuk dönteni, hogy kik kapják meg az adásunkat. A micro:bit erre egy meglehetősen egyszerű megoldást alkalmaz: a rádiót különböző csatornákra állíthatjuk be (0 és 83 közt). Ez ugyanúgy működik, mint a walkie-talkie-k esetében. Azok, akik ugyanarra a csatornára állnak, hallják egymást, míg mindenki más nem hallja az adott csatornán folyó beszélgetést.

3. Szentjánosbogarak

Ezúttal egy természetből vett példával demonstráljuk a rádió használatát. A szentjánosbogarak ún. biolumineszcencia segítségével jeleznek egymásnak. Rajzásuknál ezekből a jelzésekből egy nagyon látványos fényjáték áll össze. Valami ehhez hasonlót fogunk létrehozni a micro:bitek segítségével.

Első lépésként be kell importálnunk a `radio` modult a már ismert `import radio` utasítással. A rádió alapesetben energiatakarékossági okok miatt ki van kapcsolva, bekapcsolását a `radio.on()` metódussal érhetjük el. Ilyenkor a rádió alapbeállításokkal indul el, mindenféle konfiguráció nélkül használatba vehetjük, az eszközeink képesek lesznek kommunikálni egymással. Természetesen, amikor több eszközt egymástól függetlenül akarunk ilyen célokra használni, szükség lehet bizonyos beállításokra. A korábban már említett csatorna mellett sok más beállítási lehetőségünk is van, ezeket szükség esetén érdemes a dokumentáció vonatkozó oldalán[13] tanulmányozni.

Amennyiben megfelelőek számunkra az alapbeállítások, már el is kezdhetjük az üzenetek küldését:

```
radio.send("uzenet")
```

[13] https://microbit-micropython.readthedocs.io/en/latest/radio.html Elérés dátuma: 2019.01.24.

Az üzenetek fogadása is hasonlóan könnyen megy:

```
new_message = radio.receive()
```

Ahogy az üzeneteket megkapjuk, egy sor adatszerkezetbe kerülnek bele. `A receive()` metódus egy sorból műveletnek felel meg, ezáltal helyet csinál az új üzeneteknek. Ha a sor megtelik, az új bejövő üzeneteket figyelmen kívül hagyjuk.

Ezzel a két paranccsal már el is tudjuk készíteni a szentjánosbogár-animációnkat:

```
1.  import radio
2.  import random
3.  from microbit import *
4.
5.  flash = [Image().invert()*(i/9) for i in range(9, -1, -
    1)]
6.
7.  radio.on()
8.
9.  while True:
10.     if button_a.was_pressed():
11.         radio.send("flash")
12.     incoming = radio.receive()
13.     if incoming == "flash":
14.         sleep(random.randint(50, 350))
15.         display.show(flash, delay=100, wait=False)
16.         if random.randint(0, 9) == 0:
17.             sleep(500)
18.             radio.send("flash")
```

Az első három sorban elvégezzük a szükséges importálásokat. Az 5. sorban megadjuk a lejátszandó animációt. Az értelmezést nyugodtan bízzuk rá a tanulókra! Segítségként tanulmányozzák át a dokumentáció `Image`[14] aloldalát! Végül a 7. sorban bekapcsoljuk a rádiót az eszközön. Ezzel végeztünk az előkészületekkel, jöhet a program lényegi fázisa.

Természetesen megint végtelen ciklust használunk, ezzel biztosítva a folyamatos működést. A 10-11. sorban olvasható, hogy az „A” gomb megnyomására tudunk jelet küldeni. A következő sorban kiolvassuk a legrégebben megkapott jelet egy változóba, majd, ha ez megegyezik a „`flash`” szöveggel, tudjuk, hogy indulhat a villogás. Véletlen időtartamú várakozás után kijelezzük a korábban megadott animációt. Ezek után a 16. sorban eldöntjük, hogy tovább küldjük-e a jelet. Ez véletlenszerűen történik meg, jelen

[14] https://microbit-micropython.readthedocs.io/en/latest/image.html Elérés dátuma: 2019.01.24.

esetben 10% eséllyel. Így lehetséges több eszközt közt fenntartani az effektust. A rendelkezésre álló eszközök számától függően lehet a továbbküldés esélyét növelni, vagy csökkenteni. Amennyiben továbbküldésre kerül a sor, hagyjunk egy kis időt az animáció végigfutására, majd elküldjük a „flash” üzenetet.

Végeredményben legalább 9-10 micro:bittel egy nagyon látványos effektust hozhatunk létre, főleg egy sötét teremben. A véletlenszerű villogás, elhalványulás a szentjánosbogarak rajzására hasonlít.

4. Hideg-meleg játék

Bár a rádió funkció használatával egyszerre több eszközre is szükségünk volt a programjaink működéséhez, minden egyes eszközön ugyanaz a kód futott. Ezúttal egy olyan programot fogunk írni, aminél a különböző eszközökre különböző kód is kerül. Kiskorában biztosan mindenki játszott a szüleivel, esetleg testvéreivel, barátaival hideg-meleg játékot. Ennek a lényege, hogy elrejtünk egy tárgyat, majd valaki annak keresésére indul. A tárgyat elrejtő személy a hideg-langyos-meleg szavakkal jelzi a keresőnek, hogy milyen messze, vagy közel jár a tárgy megtalálásához. A micro:bitek segítségével fogjuk újra alkotni ezt a gyerekkori játékot. Jelen esetben az elrejtett tárgy egy micro:bit lesz, ami rádiójeleket fog sugározni magából. A keresőnél lesz egy másik micro:bit, ami ezeket a jeleket fogja, és a jelerősség alapján különböző ikonokat jelenít meg a kijelzőn, attól függően, hogy mennyire vagyunk közel a másik eszközhöz. A teljes megoldás azonnali ismertetése helyett ezúttal lépésenként fogjuk felépíteni az ehhez szükséges programokat, teendőket.

Az első lépés tehát a jeladó kódjának megírása. Ezzel nagyon egyszerű dolgunk lesz, hiszen az eszköz semmi mást nem fog csinálni, mint egy folyamatos rádiójelet sugározni. A szükséges beállítások elvégzése után tehát csak pár sort kell írnunk.

```
1. import radio
2.
3. radio.config(power=4)
4. radio.on()
5.
6. while True:
7.     radio.send("0")
```

A rádiónál már megszokott importálás és bekapcsolás között feltűnik egy új művelet, a `radio.config()`. Ennek a metódusnak a paramétereiben állíthatjuk be az eszköz

rádiójának különféle tulajdonságait, például a már korábban tárgyalt csatornát. Ezúttal a jelerősséget állítjuk be, hogy egy kisebb teremben és megfelelő különbségeket lássunk az adó eszköztől távol és közel mért értékek között. Ezt a `power` paraméter értékének változtatásával érhetjük el. Ez egy 0 és 7 között lévő érték lehet, az alapértelmezett a 6-os. Ezt tehát valamivel gyengébbre vettük. Amelyik paraméterekre nem térünk ki, az az alapértelmezett értékekkel fog szerepelni, ezért is szükséges a paraméter nevét is specifikálni, és egy egyenlőségjel után megadni a kívánt értéket. Eddig egyáltalán nem hívtuk meg a `config()` metódust, ami azt jelenti, hogy minden tulajdonságból az alapértelmezett értékek lesznek használva. Azt, hogy melyek ezek, a dokumentációban[15] tudjuk ellenőrizni. Ezután jön a már megszokott végtelen ciklus, hiszen a jel sugárzására folyamatosan szükségünk van. Ezen belül már csak egy üzenetet kell küldenünk, aminek a szövege tetszőleges lehet. A lényeg, hogy ugyanezt az üzenetet kell majd figyelnünk a vevőkkel. Ebben a pár sorban végeztünk is az adó kódjával, tökéletesen fog működni a projektünk. Azonban célszerű még egy kis szépítést végrehajtanunk! Jelenleg, ha elindítjuk a micro:bitet, csak egy sötét kijelzőt látunk. Érdemes valamiféle animációval kísérni a működést a kijelzőn, hogy megbizonyosodjunk róla, hogy a ciklus valóban fut, és az eszköz folyamatosan küldi a jeleket. Egészítsük hát ki a programot ezzel!

```
1.  import radio
2.
3.  radio.config(power=4)
4.  radio.on()
5.
6.  while True:
7.      radio.send("0")
8.      display.show(Image.HEART_SMALL)
9.      sleep(500)
10.     display.show(Image.HEART)
11.     sleep(500)
```

A rádiójel küldése után egy szívdobbanás animációt fogunk szimulálni, ezzel jelezve, hogy épp működésben van az eszköz. Először megjelenítjük a beépített ikonkészlet közül a kis szívet, majd fél másodperc várakoztatás után átváltunk a nagyra. Újabb fél másodperc után újraindul a ciklus, egy új jel küldésével, majd újból a kis szív megjelenítésével. Működés közben a két kép váltakozása olyan animációt hoz létre, mintha dobogna az eszköz „szíve”. Ebben a megoldásban körülbelül egy

[15] https://microbit-micropython.readthedocs.io/en/latest/radio.html Elérés dátuma: 2019. 02. 04.

másodpercenként küldjük a jeleket, majd a vevő használatánál ezt figyelembe kell vennünk.

Ezzel az adónk készenáll, töltsük rá a fenti programot, majd tegyük félre. Következhet a vevő programjának a megírása. Át kell gondolnunk, hogy hogyan állapíthatnánk meg egy rádiójelről, hogy mennyire van közel. Szerencsére a MicroPythonban egy üzenet fogadásánál elérhetjük annak a jelnek az erősségét is. A blokkos programozásnál ez egy nagyon egyszerűen elérhető érték volt. Sajnos a Pythonos környezetben nincs rá egy kész függvény, kicsit kutakodni kell a dokumentációban, valamint a Python nyelvet is ismerni kell. Azt már tudjuk, hogy az üzenet fogadása a `radio.receive()` metódussal történik, ebből azonban nem tudjuk megállapítani a jelerősséget. Ehhez egy másik fajta üzenetfogadási módot, a `radio.receive_full()` metódust kell használnunk. Ez egy ún. tuple-t ad vissza, aminek magyar nyelvű megfelelője nem igazán van, talán egy rekordra, vagy listára hasonlít leginkább. Egy tuple-ben több érték található, vesszővel elválasztva. Különbség az előbb felsoroltakhoz képest, hogy értékeit nem tudjuk megváltoztatni. Amennyiben más értékekre volna szükségünk egy tuple-ben, újat kell létrehoznunk. Amiben viszont hasonlít hozzájuk, hogy egyes elemeire indexeléssel hivatkozhatunk, az indexet pedig szögletes zárójelek között adhatjuk meg. További információk a tuple-ről a Python dokumentációjában[16] találhatóak, itt csak a feladathoz szükséges mértékben jelennek meg a részletek.

A `radio.receive_full()` metódus egy három elemű tuple-t ad vissza. Ennek az elemei sorban:

- a következő bejövő üzenet az üzeneteket tartalmazó sorban, bájtként
- a jelerősség, ami egy 0 (legerősebb) és -255 (leggyengébb) között lévő egész szám, a jel erőssége dBm-ben (decibel milliwattban)
- egy időbélyeg

Ezek alapján elkezdhetjük írni a vevő kódját. Először teszteljük le, hogy fogjuk-e a jelet, és meg tudjuk-e állapítani annak erősségét. Folyamatosan írjuk ki ezt a kijelzőre!

[16] https://docs.python.org/3/tutorial/datastructures.html#tuples-and-sequences Elérés dátuma: 2019. 02. 02.

```
1. import radio
2.
3. radio.on()
4.
5. while True:
6.     message = radio.receive_full()
7.     if message:
8.         display.scroll(message[1])
```

A program első sorai a már megszokottak. A lényegi részben használjuk a fentebb megismert üzenetfogadási módot. A 7. sorban található ellenőrzésre mindenképpen szükségünk van, enélkül hibát dobna a program, mert nem biztos abban, hogy egy létező értékre hivatkozunk a 8. sorban. Ebben láthatjuk, hogy megjelenítjük a kijelzőn a kapott tuple 2. elemét, ami az 1-es indexű, hiszen a Pythonban is 0-ás kezdőindexet alkalmazunk, akár a legtöbb modern programozási nyelvben.

Ezzel kész az ellenőrző programunk, teszteljük le a működést! Töltsük rá a vevő eszközre a programot, majd kapcsoljuk be az adót, akár a számítógéphez csatlakoztatva, akár elemről működtetve. Az adón megindul a szívdobogás, ezzel párhuzamosan a vevőnek ki kell írnia a kapott jelek erősségét. Végezzünk pár próbamérést, különböző távolságokból. Döntsük el, hogy mik legyenek a küszöbértékek a hideg, a meleg, illetve a forró jelzésekhez, majd jegyezzük fel ezeket! Ezek nagyban függenek a mérés helyétől és az esetleges zavaró körülményektől, így az itt szereplő értékek csupán példaként szolgálnak, minden helyszínen saját mérést kell végezni. Esetemben -75 dBm alatt lesz a hideg, -65 és -75 dBm között a meleg, és -65 dBm felett pedig a forró jelzése.

Ezután módosítsuk a vevők programját! A kalibrációval végeztünk, így a jelerősségek kiírására már nincs szükségünk. Ehelyett a meghatározott küszöbértékek mentén kell egy megfelelő ikont kijelezni a vevő kijelzőjén. Emlékezzünk vissza az első alkalommal kipróbált `Image` modul képeire! A tanulókkal együtt keressünk három megfelelő ikont, ami viszonylag egyértelműen jelzi, hogy a hideg, a meleg, vagy a forró tartományában járunk! A diákok gondolják át, hogy milyen vezérlési szerkezetre lesz itt szükség! Valószínűleg rá fognak jönni, hogy az elágazásra, de vajon hány feltétel lesz benne? Első válaszként várhatóan hármat fognak mondani, hiszen három intervallumunk van, az ezekbe való beletartozást kell vizsgálnunk. Azonban vezessük rá őket, hogy kettő vizsgálat is elég, hiszen hogyha az első két intervallumba nem tartozik egy érték, akkor biztosak lehetünk benne, hogy a harmadikban benne van! A programot ezek alapján már önálló munkában is el tudják készíteni a diákok. A javasolt megoldás:

```
1.  import radio
2.
3.  hot = Image("99999:99999:99999:99999:99999")
4.
5.  radio.on()
6.
7.  while True:
8.      message = radio.receive_full()
9.      if message:
10.         if message[1] < -75:
11.             display.show(Image.DIAMOND_SMALL)
12.         elif message[1] < -65:
13.             display.show(Image.DIAMOND)
14.         else:
15.             display.show(hot)
```

Az egyes értékekhez tartozó képek kiválasztásánál én a kis gyémánt, a gyémánt, és az összes LED felvillantása mellett döntöttem. Ez utóbbit a 3. sorban deklaráltam is, a másik kettő beépített kép, így ezek előzetes felvételére nincs szükség. A rádió bekapcsolása után kezdődhet is a végtelen ciklus, hiszen az éppen aktuális „hőmérséklet” kiírása folyamatosan szükséges. A jelerősséget a kalibrálásnál megismert módon olvassuk ki az üzenetből, majd jöhet az elágazás. Először ellenőrizzük, hogy a hideg tartományban van-e a mért érték. Amennyiben igen, a kijelzőn megjelenítjük a hozzá tartozó képet. Ezután a meleg tartomány következik. Figyeljük meg, hogy a feltételben nem az szerepel, hogy az érték nagyobb mint -75 és kisebb mint -65, elég csak az utóbbit ellenőrizni, hiszen az első feltétel nem teljesüléséből már következik, hogy a jelerősség nagyobb mint -75. Ezért fontos, hogy mindenképpen az `elif` kulcsszót használjuk, különben sosem látnánk a kijelzőn a hideget jelképező kis gyémántot, hiszen ami -75-nél kisebb, az -65-nél is. Legvégül, ha ezek egyike sem teljesül, biztosak lehetünk benne, hogy a jelerősség legalább -65 dBm, így megjeleníthetjük a forró tartományhoz tartozó képet.

A vevő programja ezzel el is készült, eljött a kipróbálás ideje! A diákok páronként az egyikük eszközére az adó, a másikuk eszközére pedig a vevő kódját töltsék fel, ügyelve a csatornák helyes beállítására. Ne legyen egynél több pár ugyanazon a csatornán! Ezután csatlakoztassák az elemeket vagy egy-egy powerbanket a micro:bitekhez, egy rövid teszttel bizonyosodjunk meg róla, hogy működnek a programok, majd az egyik diák dugja el a teremben, a folyosón, vagy az iskola egy meghatározott részén az adót. A másik diák induljon el a vevővel, és próbálja megkeresni azt! Ezután cserélődjenek fel a szerepek, dugja el a másik diák az adót, és a párja keresse meg!