Micro:bit – pierwsze kroki z Pythonem

Autorzy: Dawid Osuchowski, Filip Kłębczyk

Lekcja 6:

Akcelerometr

W ramach zajęć uczniowie zapoznają się z podstawową obsługą akcelerometru (wykrywanie gestów). Stworzą program wyświetlający prostą animację pulsującego serca z możliwością zmiany tempa pulsu dzięki wykorzystaniu gestów.

Cele zajęć:
Uczeń powinien:

- Znać pojęcia: akcelerometr, gest;
- Wiedzieć, jakie zastosowania ma akcelerometr;
- Potrafić odczytywać gesty przy użyciu akcelerometru.

Pojęcia kluczowe:
akcelerometr, gest, operator not

Metody pracy:

- Wykład, dyskusja, prowadzenie
- Ćwiczenia praktyczne przy komputerze

Materiały pomocnicze:

- http://microbit.org/
- https://bbcmicrobitmicropython.readthedocs.io/
- https://github.com/MicrobitPolska/FunWithMicrobit

Czas na realizację zajęć: 45 min

Treści programowe (związek z podstawą programową)
Podstawa programowa z Informatyki dla szkół podstawowych, klasy VII-VIII

- I Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
  1. formułuje problem w postaci specyfikacji (czyli opisuje dane i wyniki) i wyróżnia kroki w algorytmicznym rozwiązywaniu problemów. Stosuje różne sposoby przedstawiania algorytmów, w tym w języku naturalnym, w postaci schematów blokowych, listy kroków;
  3. przedstawia sposoby reprezentowania w komputerze wartości logicznych, liczb naturalnych (system binarny), znaków (kody ASCII) i tekstów;
  4. rozwija znajomość algorytmów i wykonuje eksperymenty z algorytmami, korzystając z pomocy dydaktycznych lub dostępnego oprogramowania do demonstracji działania algorytmów;
  5. prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów.
- II Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Uczeń:
  1. projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje oraz zmienne i tablice. W szczególności programuje algorytmy z działu I pkt 2;
  2. projektuje, tworzy i testuje oprogramowanie sterujące robotem lub innym obiektem na ekranie lub w rzeczywistości;
  4. zapisuje efekty swojej pracy w różnych formatach i przygotowuje wydruki;
- III Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi. Uczeń:
  2. rozwija umiejętności korzystania z różnych urządzeń do tworzenia elektronicznych wersji tekstów, obrazów, dźwięków, filmów i animacji;
  3. poprawnie posługuje się terminologią związaną z informatyką i technologią.
- IV Rozwijanie kompetencji społecznych. Uczeń:
  1. bierze udział w różnych formach współpracy, jak: programowanie w parach lub w zespole, realizacja projektów, uczestnictwo w zorganizowanej grupie uczących się, projektuje, tworzy i prezentuje efekty wspólnej pracy;

Przebieg zajęć:

1. Wprowadzenie w tematykę zajęć – 10 min.

Na początku zajęć przypominamy sobie, jakie funkcje były wykorzystywane do obsługi wyświetlacza. Następnie przechodzimy do omówienia elementu akcelerometru i jego zastosowania. Pytamy uczniów, jakie jeszcze urządzenia zawierające akcelerometr znają (telefon komórkowy, tablet, smartwatch).

2. Część zasadnicza – 30 min.

Pierwszy program, który stworzymy w ramach zajęć, będzie wykorzystywał gest obrotu płytki Micro:bit w prawo. W momencie gdy obrócimy płytkę w prawą stronę, wyświetlacz pokaże nam strzałkę w prawo, w przeciwnym razie nie wyświetli nic.

```
1  from microbit import *
2
3  while True:
4      gesture = accelerometer.current_gesture() # odczytujemy aktualny gest, jaki
   zarejestrowała płytka
5      if gesture == "right": # sprawdzamy czy płytka była obrócona w prawo
6          display.show(Image.ARROW_E)
7      else:
8          display.show(" ")
```

Teraz uczniowie powinni spróbować rozbudować go, aby obsługiwał również gest obrotu w lewo. Wyświetlacz w momencie obrotu w lewo wyświetli strzałkę w lewo.

```
1  from microbit import *
2
3  while True:
4      gesture = accelerometer.current_gesture()
5      if gesture == "right":
6          display.show(Image.ARROW_E)
7      # napisz tutaj kod, który wyświetli strzałkę w lewo wtedy,
```

```
    # kiedy przechylimy płytkę w lewo
    else:
        display.show(" ")
```

Rozwiązanie:

```
from microbit import *

while True:
    gesture = accelerometer.current_gesture()
    if gesture == "right":
        display.show(Image.ARROW_E)
    elif gesture == "left": # sprawdzamy czy płytka była obrócona w lewo
        display.show(Image.ARROW_W)
    else:
        display.show(" ")
```

Kolejnym programem, który zbudujemy w ramach lekcji, będzie pulsujące na wyświetlaczu serce. Prędkość pulsowania będzie można zmieniać poprzez odchylanie płytki w lewo i prawo.

```
from microbit import *

delay = 250

while True:
    gesture = accelerometer.current_gesture()
    if gesture == "right" and delay > 50:
        delay -= 100
    elif gesture == "left" and delay < 550:
        delay += 100
    display.show(Image.HEART)
    sleep(delay)
    display.show(Image.HEART_SMALL)
    sleep(delay)
```

Chcemy, by w programie nasze opóźnienie było nie mniejsze niż 50 ms i nie większe niż 550 ms. W tym celu w instrukcji warunkowej sprawdzamy nie tylko, czy dany gest został wykonany, ale również, czy opóźnienie znajduje

się w pożądanym przez nas zakresie. Jeśli oba warunki są spełnione (operator and), to odpowiednio zwiększamy lub zmniejszamy opóźnienie.

Kolejny program jest ulepszeniem poprzedniego, jedyną zmianą jest dodanie dodatkowej instrukcji warunkowej, która pozwoli nam zlikwidować dwa wywołania funkcji sleep do jednego.

```
1  from microbit import *
2
3  delay = 500
4  small = False
5
6  while True:
7      gesture = accelerometer.current_gesture()
8      if gesture == "right" and delay > 100:
9          delay -= 100
10     elif gesture == "left" and delay < 1000:
11         delay += 100
12
13     if small:
14         display.show(Image.HEART_SMALL)
15     else:
16         display.show(Image.HEART)
17     small = not small
18     sleep(delay)
```

W linii 17 tworzymy zmienną pomocniczą small, która posłuży nam do przełączania obrazka po każdym obiegu pętli. Korzystając z operatora not zmieniamy jej wartość pomiędzy wartościami logicznymi True i False (prawda i fałsz) w kolejnych obiegach pętli. W rezultacie, w zależności czy obieg pętli jest parzysty, czy nieparzysty, ustawiamy określony obrazek dużego lub małego serca.

3. Podsumowanie – 5 min.

W ramach podsumowania dyskutujemy z uczniami na temat zrealizowanych rozwiązań i zachęcamy ich do zapoznania się z innymi gestami obsługiwanymi przez płytkę Micro:bit.