

Micro:bit – pierwsze kroki z Pythonem

Autorzy: Dawid Osuchowski, Filip Kłębczyk

Lekcja 8:

Morse z radiem i brzęczykiem

W ramach zajęć, w oparciu o zdobytą wcześniej wiedzę, uczniowie zrealizują dwa programy – nadajnika i odbiornika sygnałów w kodzie Morse’a. Ponadto zapoznają się i zastosują w praktyce kolejny rodzaj pętli – pętlę for, której użyją w programie odtwarzającym sygnały dźwiękowe.

Cele zajęć:

Uczeń powinien:

- Znać pojęcia: alfabet Morse’a, radio, brzęczyk, format danych, pętla for;
- Wiedzieć, jakie zastosowanie ma pętla for w języku Python i jak jej użyć;
- Potrafi definiować prosty format wymiany danych.

Pojęcia kluczowe:

alfabet Morse’a, radio, brzęczyk, format danych, pętla for

Metody pracy:

- Wykład, dyskusja, prowadzenie
- Ćwiczenia praktyczne przy komputerze
- Burza mózgów
- Szukanie rozwiązań na postawiony problem

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.

Ukończenie tej lekcji wymaga znajomości działania alfabetu Morse’a, dlatego pytamy uczniów, czy kiedykolwiek się z nim zetknęli. Jeżeli nie, to krótko przedstawiamy genezę kodu oraz jego podstawowe zasady i zastosowania. Powinniśmy również przypomnieć uczniom informacje związane z obsługą brzęczyka oraz radia na Micro:bitcie.

Nauczyciel dzieli uczniów na grupy, w połowie grup płytka Micro:bit będzie służyć jako nadajnik, a w drugiej połowie jako odbiornik.

W zależności od posiadanego brzęczyka, jeśli ten nie reaguje na zmiany stanu na PINie po wywołaniu metody `write_digital` na `pin0`, to należy zastosować prostą podmianę w poniższych programach:

<code>pin0.write_digital(1)</code>	<code>music.pitch(440)</code>
<code>pin0.write_digital(0)</code>	<code>music.stop()</code>

W takim przypadku należy też zaimportować moduł `music`.

2. Część zasadnicza – 30 min.

W pierwszej wersji programu nadajnik po naciśnięciu przycisku będzie wysyłał poprzez radio sygnał SOS, odbiornik odbiera ten sygnał i za pomocą brzęczyka przedstawia go w postaci alfabetu Morse’a. Zanim uczniowie zabiorą się do pisania kodu, muszą wspólnie wymyślić format danych, za pomocą którego będą przekazywać sygnały między płytkami. W alfabecie Morse’a występują tylko dwa sygnały, krótki i długi. Ich zadanie polega więc na tym, aby w jakiś sposób rozróżnić te sygnały między sobą. Przykładowy format, który możemy zaproponować, będzie składać się wyłącznie z zer i jedynek. Przyjmujemy, że cyfra 0 będzie oznaczać sygnał krótki, a 1 sygnał długi. Nasz przykładowy kod podczas tej lekcji będzie korzystał z tej konwencji.

Kod nadajnika:

```
1 import radio
2 from microbit import *
3
4 radio.config(channel=5)
5 radio.on() # włączenie radio
6
7 while True:
8     if button_a.was_pressed():
9         radio.send('000111000') # wysłanie sygnału SOS
10
11     display.show('SOS')
```

W poleceniu `radio.send()` zapisujemy sygnał SOS zgodnie z formatem, jaki przyjęliśmy do komunikacji pomiędzy płytkami. Litera S w alfabecie Morse'a składa się z trzech krótkich sygnałów, natomiast litera O z trzech długich sygnałów.

Teraz zajmiemy się napisaniem kodu odbiornika, który będzie odtwarzał dowolny, przesłany ciąg zer i jedynek na brzęczyku. Otrzymany program posłuży nam również później, gdy rozbudujemy kod nadajnika.

```
1 import radio
2 from microbit import *
3
4 unit = 50
5
6 radio.config(channel=5) # konfiguracja radia
7 radio.on() # włączenie radio
8
9 while True:
10     code = radio.receive() # nasłuchiwanie i odbiór danych
11     if code: # sprawdzenie czy coś zostało przesłane
12         for signal in code:
13             pin0.write_digital(1)
14             if signal == "0": # sygnał krótki
15                 sleep(unit)
16             elif signal == "1": # sygnał długi
17                 sleep(3 * unit)
```

```
18     pin0.write_digital(0)
19     sleep(unit)
```

W programie pojawia się nowy rodzaj pętli – for. Pętla for w Pythonie jest wykorzystywana do wykonywania kolejnych kroków na elementach niektórych typów (np. list), ciągów znakowych lub specjalnych rodzajach funkcji określanych generatorami (np. range()).

Przykładowo, by wykonać pętlę określoną liczbę razy (w poniższym przykładzie 5) napiszemy:

```
for i in range(5):
    display.show(i)
    sleep(500)
```

Gdy będziemy chcieli w kolejnych krokach pętli uzyskiwać poszczególne znaki z ciągu znakowego, napiszemy:

```
for c in "Tekst":
    display.show(c)
    sleep(500)
```

W kodzie odbiornika w pętli for `signal in code` zmienna `signal` będzie przyjmowała kolejne znaki z ciągu `code`. Przy pomocy instrukcji warunkowych sprawdzamy otrzymany komunikat znak po znaku i w zależności od niego odtwarzamy krótki lub długi sygnał dźwiękowy. Dodatkowo dla wygody definiujemy zmienną `unit`, która przechowuje czas trwania pojedynczego, krótkiego sygnału. W alfabecie Morse'a długi sygnał powinien być trzykrotnie dłuższy ($3 \cdot \text{unit}$) od sygnału krótkiego (`unit`). Z kolei przerwa między kolejnymi sygnałami trwa dokładnie ten sam okres czasu co sygnał krótki (`unit`).

Finalnie zmieniamy program nadajnika, który zamiast jednego konkretnego sygnału SOS, będzie umożliwiał wybór litery alfabetu przyciskiem A, a następnie umożliwiał jej przesłanie po naciśnięciu przycisku B odpowiadającym jej kodem Morse'a.

```
1 import radio
2 from microbit import *
3
4 index = 0
5 letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
6 morse_code = ['01', '1000', '1010', '100', '0', '0010', '110', '0000', '00',
7              '0111', '101', '0100', '11', '10', '111', '0110', '1101', '010',
8              '000', '1', '001', '0001', '011', '1001', '1011', '1100']
9
10 radio.config(channel=5)
11 radio.on() # włączenie radio
12 while True:
13     if button_a.was_pressed():
14         index += 1
15         if index == len(letters):
16             index = 0
17     elif button_b.was_pressed():
18         radio.send(morse_code[index]) # wysłanie sygnału przez radio
19     display.show(letters[index])
```

Zmienna `letters` przechowuje dwadzieścia sześć liter alfabetu. Z kolei lista `morse_code` zawiera odpowiadające im kombinacje sygnałów według alfabetu Morse'a w wybranym przez nas wcześniej formacie danych.

3. Podsumowanie – 5 min.

Pod koniec zajęć dyskutujemy z uczniami na temat ograniczeń zrealizowanych w ramach zajęć programów (np. wysyłanie tylko jednej litery) i zastanawiamy się, jak można by było rozbudować istniejący kod.

4. Alternatywy

Alternatywnym projektem, który możemy stworzyć, używając poznanych do tej pory funkcji oraz elementów, jest zdalnie kontrolowany odtwarzacz muzyczny. Jedna z płytek pełnić będzie rolę odtwarzacza, do którego podłączony jest brzęczyk, druga natomiast obejmie funkcję pilota zdalnego sterowania radiowo. W przypadku Micro:bita 2 obie płytki mają brzęczyk, więc ta która będzie pełniła rolę pilota nie będzie go wykorzystywała.

Najlepiej będzie zacząć od napisania kodu pilota, gdyż nie jest on bardzo skomplikowany.

```
1 # Kod pilota
2 import radio
3 from microbit import *
4
5 radio.config(channel=1) # konfiguracja radia
6 radio.on()
7
8 while True:
9     if button_a.was_pressed():
10         radio.send("play/stop") # wysłanie sygnału rozpoczynającego lub kończącego odtwarzanie
11     elif button_b.was_pressed():
12         radio.send("next") # wysłanie sygnału zmieniającego piosenkę
```

Nasz pilot ma dwie funkcje obsługiwane przyciskami A i B. Przycisk A pełni funkcję Play/Stop tzn. służy do odtwarzania i zatrzymywania muzyki, natomiast przycisk B pozwala zmieniać odtwarzaną melodyjkę. W zależności, który przycisk zostanie naciśnięty, pilot wyśle poprzez radio odpowiadający jego funkcji ciąg znaków "play/stop" lub "next".

Na drugim urządzeniu, które pełni rolę odtwarzacza, będą realizowane następujące zadania – odbiór sygnału z pilota i odpowiednie działanie, w zależności jaki sygnał został przesłany. Ponadto urządzenia wyświetla numer aktualnie wybranej melodyjki (niezależnie, czy jest w danej chwili odtwarzana czy nie). Melodyjki są odtwarzane z wykorzystaniem modułu music.

```
1 # Kod odtwarzacza
2 import radio, music
3 from microbit import *
4
5 songs = [music.BLUES, music.NYAN, music.FUNERAL]
6 song_number = -1
7 play = False
8
9 incoming_data = 'next'
10
11 radio.on()
12
13 while True:
14     if incoming_data:
15         if incoming_data == 'play/stop': # po kliknięciu play/stop na pilocie
```

```
16     play = not play # przełączamy się między play a stop
17     elif incoming_data == 'next': # po kliknięciu next na pilocie
18         song_number += 1
19         if song_number == len(songs): # kiedy index wykracza poza zakres
20             song_number = 0 # resetujemy go, żeby nie wywołać błędu
21             display.scroll(song_number + 1, wait=False, loop=True)
22         if play:
23             music.play(songs[song_number], wait=False, loop=True) # słowo kluczowe
                'wait' jest potrzebne, aby nie blokować reszty programu, dzięki niemu muzyka gra w
                tle
24         else:
25             music.stop()
26     incoming_data = radio.receive()
```

W powyższym kodzie wykorzystywane są cztery zmienne. Są to:

- songs – lista z melodyjkami,
- song_number – indeks aktualnej melodyjki z listy,
- play – przyjmuje wartość True lub False odpowiadającą stanowi urządzenia – odtwarzanie lub nie melodyjki,
- incoming_data – przechowuje odebrany sygnał z pilota (początkowo przyjmuje wartość “next” po to, by po uruchomieniu została wywołana funkcja display i wybrana pierwsza pozycja z listy).

Kod odtwarzacza można też zmodyfikować i zamiast funkcji stop realizować funkcję pauzy (wznawianie odtwarzania melodyjki od momentu zatrzymania). Tego typu zadanie można zadać uczniom ambitnym, gdyż jego realizacja nie jest prosta.