Mikrobitowcy

Autorzy: Dawid Osuchowski, Filip Kłębczyk

Lekcja 12:
Micro:bit - kod Morse'a z radiem i brzęczykiem

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, którą użyją w programie odtwarzającym sygnały dźwiękowe.

Cele lekcji:
Uczeń powinien:
- Znać pojęcia: alfabet Morse'a, radio, brzęczyk, format danych, pętla for
- Wiedzieć, jakie zastosowania ma pętla for w jzyku Python i jak jej użyć
- Potrafi definiować prosty format wymiany danych

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

Pojęcia kluczowe:
→ alfabet Morse'a → radio → brzęczyk → format danych → pętla for

Czas realizacji: 45 min.

Metody pracy:
- wykład,
- dyskusja,
- ćwiczenia praktyczne przy komputerze,
- szukanie rozwiązań na postawiony problem
- burza mózgów.

Treści programowe:
Podstawa programowa kształcenia ogólnego dla szkół podstawowych – II etap edukacyjny – klasy VII-VIII, informatyka:
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;
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;

Wprowadzenie w tematykę i integracja grupy (10 min.)

Ukończenie tej lekcji wymaga znajomości działania alfabetu Morse'a - nauczyciel pyta uczniów, czy kiedykolwiek się z nim zetknęli. Jeżeli nie, krótko przedstawia genezę kodu oraz jego podstawowe zasady i zastosowania. Powinniśmy również przypomnieć uczniom informacje związane z obsługą brzęczyka oraz radia na płytce Micro:bit.

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 wywoływaniu metody **write_digital** na pin0, należy zastosować prostą podmianę w poniższych programach, wg wzoru:

`pin0.write_digital(1)`	`music.pitch(440)`
`pin0.write_digital(0)`	`music.stop()`

W takim przypadku należy też zaimportować moduł **music**.

Część zasadnicza (30 min.)

Pierwsza wersja programu:

W tej wersji programu nadajnik po naciśnięciu przycisku będzie wysyłał poprzez radio sygnał SOS, odbiornik odbierze ten sygnał i za pomocą brzęczyka przedstawi 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. Zadanie uczniów polega więc na tym, aby w jakiś sposób rozróżnić te sygnały między sobą.

Przykładowy format, jaki 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 dla odbiornika, który będzie odtwarzał dowolny, przesłany ciąg zer i jedynek, na brzęczyku. Otrzymany program posłuży nam również później, gdy rozbudujemy kod nadajnika.

Wprowadzenie w tematykę i integracja grupy (10 min.)

Ukończenie tej lekcji wymaga znajomości działania alfabetu Morse'a - nauczyciel pyta uczniów, czy kiedykolwiek się z nim zetknęli. Jeżeli nie, krótko przedstawia genezę kodu oraz jego podstawowe zasady i zastosowania. Powinniśmy również przypomnieć uczniom informacje związane z obsługą brzęczyka oraz radia na płytce Micro:bit.

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 wywoływaniu metody **write_digital** na pin0, należy zastosować prostą podmianę w poniższych programach, wg wzoru:

`pin0.write_digital(1)`	`music.pitch(440)`
`pin0.write_digital(0)`	`music.stop()`

W takim przypadku należy też zaimportować moduł **music**.

Część zasadnicza (30 min.)

Pierwsza wersja programu:

W tej wersji programu nadajnik po naciśnięciu przycisku będzie wysyłał poprzez radio sygnał SOS, odbiornik odbierze ten sygnał i za pomocą brzęczyka przedstawi 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. Zadanie uczniów polega więc na tym, aby w jakiś sposób rozróżnić te sygnały między sobą.

Przykładowy format, jaki 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 dla odbiornika, który będzie odtwarzał dowolny, przesłany ciąg zer i jedynek, na brzęczyku. Otrzymany program posłuży nam również później, gdy rozbudujemy kod nadajnika.

```
import radio
from microbit import *

unit = 50

radio.config(channel=5) # konfiguracja radia
radio.on() # włączenie radio

while True:
    code = radio.receive() # nasłuchiwanie i odbiór danych
    if code: # sprawdzenie czy coś zostało przesłane
        for signal in code:
            pin0.write_digital(1)
            if signal == "0": # sygnał krótki
                sleep(unit)
            elif signal == "1": # sygnał długi
                sleep(3 * unit)
            pin0.write_digital(0)
            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*unit) od sygnału krótkiego (unit). Z kolei przerwa między sygnałami trwa dokładnie ten sam okres czasu, co sygnał krótki.

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

```
import radio
from microbit import *

index = 0
letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
morse_code = ['01', '1000', '1010', '100', '0', '0010', '110', '0000', '00',
              '0111', '101', '0100', '11', '10', '111', '0110', '1101', '010',
              '000', '1', '001', '0001', '011', '1001', '1011', '1100']

radio.config(channel=5)
radio.on() # włączenie radio
while True:
    if button_a.was_pressed():
        index += 1
        if index == len(letters):
            index = 0
    elif button_b.was_pressed():
        radio.send(morse_code[index]) # wysłanie sygnału przez radio
    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.

Podsumowanie (5 min.)

Na koniec zajęć nauczyciel inicjuje dyskusję na temat ograniczeń zrealizowanych w ramach zajęć programów (np. wysyłanie tylko jednej litery). Uczniowie wspólnie zastanawiają się, jak można by było rozbudować istniejący kod.

Alternatywy

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

```
# Kod pilota
import radio
from microbit import *

radio.config(channel=1) # konfiguracja radia
radio.on()

while True:
    if button_a.was_pressed():
        radio.send("play/stop") # wysłanie sygnału rozpoczynającego lub
kończącego odtwarzanie
    elif button_b.was_pressed():
        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 melodię. 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 urządzenie wyświetla numer aktualnie wybranej melodii (niezależnie, czy jest w danej chwili odtwarzana czy nie). Melodie są odtwarzane z wykorzystaniem modułu **music**.

```
# Kod odtwarzacza
import radio, music
from microbit import *

songs = [music.BLUES, music.NYAN, music.FUNERAL]
song_number = -1
play = False

incoming_data = 'next'

radio.on()

while True:
    if incoming_data:
        if incoming_data == 'play/stop': # po kliknięciu play/stop na pilocie
            play = not play # przełączamy się między play a stop
        elif incoming_data == 'next': # po kliknięciu next na pilocie
            song_number += 1
            if song_number == len(songs): # kiedy index wykracza poza zakres
                song_number = 0 # resetujemy go, żeby nie wywołać błędu
            display.scroll(song_number + 1, wait=False, loop=True)
        if play:
            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
        else:
            music.stop()
    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 melodii
- **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 szczególnie uzdolnionym.