Mikrobitowcy

Autorzy: Dawid Osuchowski, Filip Kłębczyk

Lekcja 9:

Micro:bit - brzęczyk

Bzzzzz! Tym razem uczniowie zapoznają się z zewnętrznym elementem elektronicznym płytki Micro:bit – brzęczykiem (buzzer piezoelektryczny). Na przykładzie tego komponentu nauczą się podłączać elementy elektroniczne do płytki oraz obsługiwać je z poziomu programów. Poza tym wrócą do tematu omawianego podczas wstępnych lekcji: operatory logiczne and, or oraz not. Wiedzę wykorzystają do stworzenia programu sterującego tempem wydawanego przez brzęczyk dźwięku.

Cele lekcji:
Uczeń powinien:
- Znać pojęcia: brzęczyk, przewód krokodylkowy, pin, masa, generowanie wartości pseudolosowych, operator logiczny, wartość logiczna
- Znać podstawowe zasady poprawnego podłączania elementów elektrycznych do płytki Micro:bit
- Umieć wydawać dźwięki z wykorzystaniem brzęczyka
- Potrafić generować wartości pseudolosowe przy pomocy modułu random
- Rozumieć różnicę i działanie operatorów logicznych and, or i
- not.

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

Pojęcia kluczowe:
→ brzęczyk → przewód krokodylkowy → pin → masa → generowanie wartości pseudolosowych → operator logiczny → wartość logiczna

Czas realizacji: 45 min.

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

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;

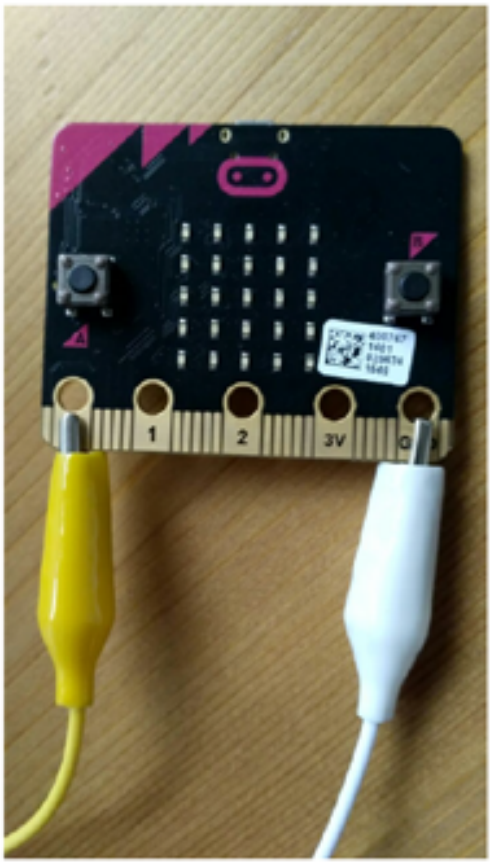
Wprowadzenie w tematykę i integracja grupy (10 min.)

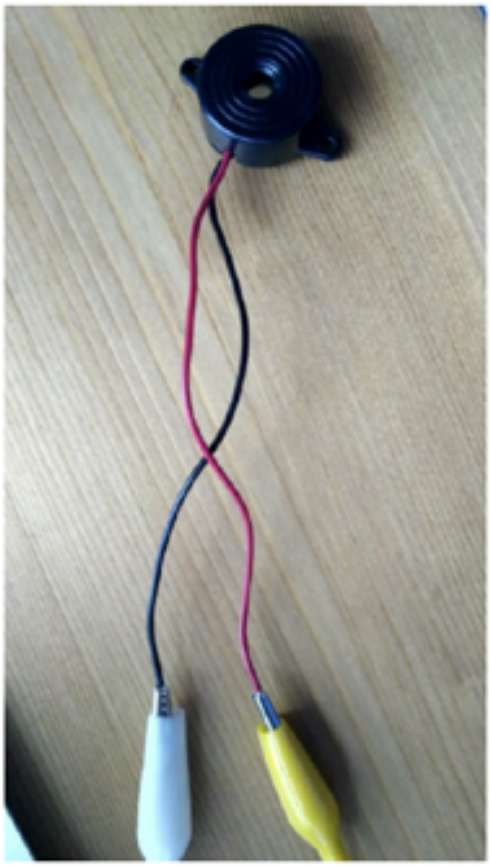
Nauczyciel przypomina uczniom zagadnienia związane z pętlą i instrukcją warunkową, które będą wykorzystywane w ramach aktualnych zajęć. Opowiada o złączach z Micro:bita i zasadach bezpiecznego podłączania elementów elektronicznych. Następnie prezentuje element typu buzzer i omawia jego funkcję, zwracając uwagę, że nie można go dowolnie podłączać. Pyta uczniów, w jaki sposób można wykorzystywać sygnały dźwiękowe.

Część zasadnicza (30 min.)

Uczniowie podłączają do swoich wyłączonych Micro:bitów brzęczyk z wykorzystaniem dwóch przewodów ze złączem krokodylkowym. W zależności od typu buzzera należy zwrócić uwagę, jak jest on podłączany do płytki.

W przypadku buzzera z przewodami czarnym i czerwonym: czarny podłączamy do przewodu podłączonego do złącza GND na Micro:bitcie (masa), a czerwony do przewodu połączonego ze złączem 0 na Micro:bitcie (PIN 0).





W przypadku buzzera z pinami dłuższym i krótszym: krótszy podłączamy przewodem do GND, a dłuższy przewodem do 0 na Micro:bitcie.

W zależności od posiadanego brzęczyka, jeśli nie reaguje na zmiany stanu na PINie po wywoływaniu metody **write_digital** na **pin0**, należy zastosować prostą podmianę w poniższych programach (w takim przypadku należy też zaimportować moduł music):

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

Na początek stworzymy najprostszy program, wydający pojedynczy sygnał dźwiękowy oczasie trwania 1 s.

```
from microbit import *

pin0.write_digital(1) # zasilamy brzęczyk poprzez pin0
sleep(1000)
pin0.write_digital(0) # odłączamy zasilanie
```

Metoda **write_digital()** wywołana na **pin0** pozwala ustawić stan pinu 0 na wysoki, co wiąże się z powstaniem różnicy napięć i przepływem prądu przez element podłączony do pinu (w przypadku brzęczyka wydanie dźwięku). Metodę możemy wywoływać również dla **pin1** oraz **pin2**.

Następnym zadaniem będzie generowanie więcej niż pojedynczego sygnału dźwiękowego. W tym celu skorzystamy z pętli:

```
from microbit import *

while True:
    pin0.write_digital(1) # zasilamy brzęczyk poprzez pin0
    sleep(50)
    pin0.write_digital(0) # odłączamy zasilanie
    sleep(1000)
```

Czas trwania pierwszej funkcji **sleep** decyduje o długości trwania dźwięku, a drugiej - o przerwie między kolejnymi dźwiękami.

Jeżeli chcemy, by nasze programy korzystały z losowości, możemy skorzystać z biblioteki **random** . Funkcja **randint()** pozwoli nam wylosować liczby, które następnie wykorzystamy do określenia długości przerw w milisekundach między dźwiękami. Przyjmuje ona dwie wartości, które definiują zakres, z którego wylosowana zostanie liczba całkowita.

```
import random
from microbit import *

while True:
    pin0.write_digital(1) # zasilamy brzęczyk poprzez pin0
    sleep(5)
    pin0.write_digital(0) # odłączamy zasilanie
    sleep(100 * random.randint(1, 10)) # losujemy liczbę z zakresu 1 do 10, a
później mnożymy ją przez 100
```

Kolejny program, jaki stworzymy w ramach tej lekcji, posłuży nam do sterowania tempem dźwięku. Do tego celu wykorzystamy wiedzę z poprzednich zajęć związaną z obsługą przycisków.

```
from microbit import *

delay = 500 # przerwa w milisekundach

while True:
    if button_a.was_pressed() and delay > 100:
        delay -= 100
    elif button_b.was_pressed() and delay < 2000:
        delay += 100
    pin0.write_digital(1) # zasilamy brzęczyk poprzez pin0
    sleep(5)
    pin0.write_digital(0) # odłączamy zasilanie
    sleep(delay)
```

W powyższym programie wykorzystujemy konstrukcję logiczną **and** w wyrażeniu sprawdzanym przez instrukcję warunkową **if**. W języku Python możemy stosować trzy operatory logiczne: **and**, **or** oraz **not**. Działają one tak, jak w innych językach programowania:

- **and** (operator logiczny "i") przyjmuje wartość prawdy, tylko gdy oba argumenty są prawdą,
- **or** (operator logiczny "lub") przyjmuje wartość prawdy, gdy przynajmniej jeden z argumentów jest prawdą,
- operator **not** (logiczne "nie") jest jednoargumentowy i neguje wartość logiczną (prawda staje się fałszem, a fałsz prawdą).

W linii 6 sprawdzamy jednocześnie, czy przycisk A był wciśnięty i czy wartość zmiennej **delay** jest większa od 100. Kod w linii 7 zostanie wykonany tylko wtedy, gdy oba warunki są spełnione. Analogicznie sytuacja ma się w przypadku linii 8 i 9.

Bardzo ambitni uczniowie, którzy szybko poradzili sobie z poprzednimi przykładami, mogą spróbować wymyślić swój własny program. Jeżeli brak im konkretnych pomysłów, sugerujemy napisanie programu realizującego funkcję metronomu. Przykładowy kod takiego programu.

```
1  from microbit import *
2
3  MS = 60000 # liczba milisekund w minucie
4  beats_per_minute = 120 # uderzenia na minute
5  display.scroll(beats_per_minute, delay=80, wait=False, loop=True)
6
7  while True:
8      if button_a.was_pressed() and beats_per_minute > 40:
9          beats_per_minute -= 10
10         display.scroll(beats_per_minute, delay=80, wait=False, loop=True)
11     elif button_b.was_pressed() and beats_per_minute < 220:
12         beats_per_minute += 10
13         display.scroll(beats_per_minute, delay=80, wait=False, loop=True)
14     pin0.write_digital(1)
15     sleep(20)
16     pin0.write_digital(0)
17     sleep(MS / beats_per_minute)
```

Podsumowanie (5 min.)

Na końcu lekcji uczniowie prezentują swoje programy nauczycielowi.

Alternatywy

Do pinów Micro:bita poza brzęczykiem możemy podłączać również inne elementy elektroniczne. Przykładowo do Micro:bita można podłączyć na podobnej zasadzie między innymi:

- Czujnik wilgotności
- Czujnik odległości
- Serwomechanizm
- Diody
- Zewnętrzne przyciski

Uwaga! Część z elementów może wymagać podłączenia dodatkowego rezystora.