Konstruktorzy gier

Autorzy: Grzegorz Zawistowski, Maciej Wojnicki

Lekcja 5-6:

Buzzer i czujnik odległości oraz własne funkcje

W pierwszej części lekcji uczniowie wykonają prosty projekt bazujący na wiadomościach zdobytych na poprzednich lekcjach. Poznają kolejne funkcje sterownika LOFI. Następnie nauczą się tworzyć własne funkcje z parametrami.

Cele lekcji:
Uczeń potrafi:
- dokonywać poprawek, zapisywać, weryfikować i wgrywać na płytkę,
- deklarować zmienną liczbową i przypisać jej wartość początkową,
- posługiwać się funkcjami write(), read(), delay(), buzzer(), distance(),
- przypisać wartość z funkcji read() i distance() zmiennej i wykorzystać ją jako wartość funkcji write(), delay() lub buzzer(), np. do sterowania czasem migania dwóch diod, czy częstotliwością buzzera,
- napisać program sterujący miganiem dwóch diod, w którym częstotliwość sterowana jest potencjometrem,
- definiować własne funkcje,
- definiować funkcje z parametrem.

Materiały pomocnicze:
- zestaw LOFI Robot CODEBOX
- komputery stacjonarne lub przenośne z zainstalowanym Arduino IDE
- komputer nauczyciela z zainstalowanym Arduino IDE, projektor, tablica projekcyjna

Pojęcia kluczowe:
→ Arduino IDE → szkic/program → otwórz, zapisz, zweryfikuj, wgraj → funkcje (setup, loop, write, delay, read, buzzer, distance) → zmienna, deklaracja funkcji, odwołanie się do funkcji → dioda → adapter LOFI Brain → potencjometr → buzzer

Czas realizacji: 90 min. (alternatywnie 45 minut - wówczas rezygnujemy z tworzenia funkcji)

Metody pracy:
- wykład problemowy,
- dyskusja dydaktyczna związana z wykładem,
- pokaz,
- ćwiczenia laboratoryjne,
- projekt.

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;
2) stosuje przy rozwiązywaniu problemów podstawowe algorytmy:
a) na liczbach naturalnych: bada podzielność liczb, wyodrębnia cyfry danej liczby, przedstawia działanie algorytmu Euklidesa w obu wersjach iteracyjnych (z odejmowaniem i z resztą z dzielenia),
4) rozwija znajomość algorytmów i wykonuje eksperymenty z algorytmami, korzystając z pomocy dydaktycznych lub dostępnego oprogramowania do demonstracji działania 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;
5) wyszukuje w sieci informacje potrzebne do realizacji wykonywanego zadania,stosując złożone postaci zapytań i korzysta z zaawansowanych możliwości wyszukiwarek.
III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi. Uczeń:
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 (5 min.)

Pytamy uczniów, co robiliśmy podczas ostatniej lekcji?

- znamy składnię kodu w środowisku Arduino IDE,
- wykorzystujemy funkcje: **write** i **read** z biblioteki LOFI,
- deklarujemy zmienne,
- piszemy proste programy sterujące miganiem diod,
- umiemy wgrywać je na płytkę Arduino UNO,
- podłączamy do płytki urządzenia wyjścia.

Dziś poznamy kolejne funkcje z biblioteki LOFI (sterowanie buzzerem i odczyt odległości) oraz *nauczymy się definiować własne funkcje. Będziemy pisać proste programy, wgrywać ja na płytkę Arduino i testować ich działanie, podłączając różne urządzenia wejścia i wyjścia

Część zasadnicza (80 min. / alternatywnie 35 min.)

Zadanie 8

Polecenie:

Napisz program sterujący dwiema diodami podłączonymi do **OUTPUT1** i **OUTPUT2**, które mają migać naprzemiennie jak pojazd uprzywilejowany. Szybkość migania diod powinna być sterowana potencjometrem podłączonym do **INPUT1**. Zapisz plik jako **“zadanie_8”**, zweryfikuj program i wgraj na płytkę.

Wskazówka 1: odczyt z potencjometru dzięki funkcji read zwraca nam zawsze jakąś wartość liczbową w zakresie od 0 do 100. Chcąc sterować “szybkością” migania diod, możemy tę zmienną wykorzystać jako parametr funkcji czekaj – delay.

Wskazówka 2: jeśli chcemy, aby diody migały wolniej, parametr funkcji delay musi być większy niż 100. Ile razy większy? 2, 3 razy? Jak to zrobić? Odpowiedź: Wprowadzając kolejną zmienną, która będzie iloczynem zmiennej pochodzącej z funkcji read oraz liczby 2, 3 lub innej.

Przykładowe rozwiązanie:

```
#include <LOFI.h>
LOFI robot;

void setup() {
}

void loop() {
int potencjometr = robot.read(INPUT1);
//definicja zmiennej potencjometr i przypisanie jej wartości z urządzenia INPUT1

int szybkosc = potencjometr * 2;
/* definicja nowej zmiennej szybkosc i przypisanie jej podwójnej
lub potrójnej wartości zmiennej potencjometr
(aby światła nie migały zbyt szybko potrzebne są nam wartości około 200,
podczas gdy maksymalny odczyt z potencjometru to 100)
*/

robot.write(OUTPUT1, 100);
delay(szybkosc); //wykorzystanie zmiennej szybkosc jako czasu świecenia diody
robot.write(OUTPUT1, 0);
robot.write(OUTPUT2, 100);
delay(szybkosc);
robot.write(OUTPUT2, 0);
}
```

Po zakończeniu zadania przechodzimy do omówienia nowego zagadnienia:

Kolejną funkcją, którą poznamy jest **buzzer(stan)**; służy do uzyskiwania sygnału dźwiękowego z wbudowanego na płytkę Arduino brzęczyka (buzzera). Parametrem funkcji buzzer jest stan, który może przyjmować wartości **true** lub **false**. Wartość **true** określa, że buzzer ma być włączony, a **false** – wyłączony.

Przykład / Ćwiczenie 9

Polecenie:

Napisz program, który będzie powodował wysyłanie krótkich sygnałów dźwiękowych. Zapisz plik jako "**zadanie_9"**, zweryfikuj program i wgraj na płytkę.

Przykładowe rozwiązanie:

```
#include <LOFI.h>
LOFI robot;

void setup() {
}

void loop() {
robot.buzzer(true);
delay(500);
robot.buzzer(false);
delay(500);
}
```

Drugą funkcją, jaką dziś poznamy jest **distance()**; służy do odczytania wartości z czujnika odległości podłączanego do specjalnego wejścia **DISTANCE**. Wartość tę możemy przypisywać do zmiennych, podobnie jak wartość odczytywaną z urządzeń wejścia podłączanych do gniazd **INPUT1-4** dzięki poleceniu **read(wejście)**.

Zadanie 10

Polecenie:

Napisz program sterujący buzzerem, tak jak czujnik parkowania w samochodzie. Częstotliwość sygnału dźwiękowego ma zależeć od odczytu odległości. Zapisz plik jako **"zadanie_10"**, zweryfikuj program, podłącz czujnik odległości do Arduino (metalowymi stykami na zewnątrz) i wgraj na płytkę.

Wskazówka: odczyt z czujnika odległości będzie zmienną, która przyjmuje wartości od 0 do 100. Chcąc uzyskać efekt powolnego piszczenia, w funkcji delay(); należy wstawić wartość około 1000 (=1 sekunda), czyli około 10 razy większą niż wartość odczytana z czujnika odległości. Jak to zrobić? Należy zdefiniować dwie zmienne. Jednej przypisać wartość odczytaną z czujnika odległości. Druga powinna być iloczynem liczby 10 i pierwszej zmiennej.

Przykładowe rozwiązanie:

```
#include <LOFI.h>
LOFI robot;

void setup() {
}

void loop() {
int odleglosc = robot.distance();
int czas = odleglosc * 10;
robot.buzzer(true);
delay(czas);
robot.buzzer(false);
delay(czas);
}
```

Na tym etapie możemy zakończyć realizację scenariusza lekcji nr 5, jeśli przeznaczyliśmy na nią 1 godzinę lekcyjną. Wówczas rezygnujemy z nauki tworzenia własnych funkcji – wszystkich poniższych treści i ćwiczeń – i przejść do podsumowania.

*Jeśli na lekcję 5. możemy przeznaczyć 2 godziny lekcyjne i zależy nam na nauce tworzenia własnych funkcji, to scenariusz realizujemy dalej:

Po wykonaniu zadania odłączamy kable USB od komputera, aby buzzery przestały piszczeć i przechodzimy do omówienia tworzenia własnych funkcji.

Mówimy uczniom: Zapewne zauważyliście, że w przypadku zaledwie kilku sygnałów świetlnych i dźwiękowych program staje się bardzo nieczytelny. Możemy jednak temu zaradzić. Chcąc uczynić nasz kod bardziej przejrzystym, można z pewnych fragmentów naszego programu, które się często powtarzają, zdefiniować odrębne własne funkcje - poza pętlą **loop();** - i wewnątrz tej funkcji odwoływać się do nich.

Oto przykład z Zadania 6:

```
#include <LOFI.h>
LOFI robot;

int czasswiecenia = 500;
int czasprzerwy = 200;

void setup() {
}

void loop() {
robot.write(OUTPUT1, 100);
delay(czasswiecenia);
robot.write(OUTPUT1, 0);
delay(czasprzerwy);
robot.write(OUTPUT1, 100);
delay(czasswiecenia);
robot.write(OUTPUT1, 0);
delay(czasprzerwy);
robot.write(OUTPUT2, 100);
delay(czasswiecenia);
robot.write(OUTPUT2, 0);
delay(czasprzerwy);
robot.write(OUTPUT2, 100);
delay(czasswiecenia);
robot.write(OUTPUT2, 0);
delay(czasprzerwy);
}
```

Jak można uprościć powyższy skrypt odpowiedzialny za miganie 2 diodek? Np. definiując nowe funkcje: jedną odpowiedzialną za mignięcie (włączenie i wyłączenie) pierwszej diody, drugą za miganie drugiej diody.

W tym celu, przed funkcją **loop()**, definiujemy własne funkcje np. **dioda1()** i **dioda2()**. Do definiowania funkcji służy polecenie **void**:

```
void dioda1(){ //definicja własnej funkcji o wymyślonej nazwie dioda1
robot.write(OUTPUT1, 100);
delay(czasswiecenia);
robot.write(OUTPUT1, 0);
}

void dioda2(){
robot.write(OUTPUT2, 100);
delay(czasswiecenia);
robot.write(OUTPUT2, 0);
}
```

Dzięki temu wewnątrz pętli **loop()** pozbędziemy się kilku powtórzeń. Zamiast tego będziemy mogli posługiwać się nazwami własnych funkcji:

```
#include <LOFI.h>
LOFI robot;

int czasswiecenia = 500;
int czasprzerwy = 200;
void setup() {
}

void dioda1(){
robot.write(OUTPUT1, 100);
delay(czasswiecenia);
robot.write(OUTPUT1, 0);
}

void dioda2(){
robot.write(OUTPUT2, 100);
delay(czasswiecenia);
robot.write(OUTPUT2, 0);
}

void loop() {
dioda1(); //odwołanie do własnej funkcji odpowiedzialnej za mignięcie 1 diody
delay(czasprzerwy);
dioda1();
delay(czasprzerwy);
dioda2();
delay(czasprzerwy);
dioda2();
delay(czasprzerwy);
}
```

Czy kod stał się bardziej przejrzysty? Uczniowie wykonują powyższe ćwiczenia i zapisują jako **“zadanie_11”**. Testują jego działanie.

Zadanie 12: Projekt RADIOWÓZ

Polecenie:

Napisz program, w którym wykorzystasz dwie diody podłączone do **OUTPUT1** i **OUTPUT2**. Będą to sygnały świetlne radiowozu. Częstotliwość migania sygnałów ma być zależna od natężenia światła (czujnik natężenia podłączony do **INPUT1**), a częstotliwość syreny alarmowej (buzzera) ma być regulowana potencjometrem podłączonym do **INPUT2**. Zapisz plik jako **“zadanie_12”**, zweryfikuj program, podłącz diody, czujnik natężenia światła i potencjometr oraz wgraj na płytkę.

Przykładowe rozwiązanie:

```
#include <LOFI.h>
LOFI robot;

void setup() {
}

void dioda1(byte czujnikswiatla) {
// definicja nowej funkcji “dioda1” z parametrem - zmienną “czujnikswiatla”
// (zmienna typu byte przyjmować może zakres od 0 do 255), odpowiadającej
// za mignięcie i zgaszenie diody nr 1,

robot.write(OUTPUT1, 100);
delay(czujnikswiatla);
robot.write(OUTPUT1, 0);
delay(czujnikswiatla);
}


void dioda2(byte czujnikswiatla) {
robot.write(OUTPUT2, 100);
delay(czujnikswiatla);
robot.write(OUTPUT2, 0);
delay(czujnikswiatla);
}


void syrena(byte czujnikswiatla) {
robot.buzzer(true);
delay(czujnikswiatla);
robot.buzzer(false);
delay(czujnikswiatla);
}

void loop() {

int czujnikswiatla = robot.read(INPUT1);

dioda1(czujnikswiatla);
// odwołanie do własnej funkcji “dioda1” z parametrem “czujnikswiatla”,
// który jest jednocześnie linijkę wcześniej zdefiniowaną zmienną odczytywaną
// przez funkcję read z wejścia INPUT1

dioda2(czujnikswiatla);
syrena(czujnikswiatla);

}
```

Podsumowanie i ewaluacja (5 min.)

Prosimy, aby uczniowie ostrożnie spakowali zestawy. Jeden przedstawiciel każdej grupy przynosi zestaw na wyznaczone przez nauczyciela miejsce w klasie.

Zadajemy uczniom pytanie: Czego nauczyliśmy się na dzisiejszej lekcji?

- wykorzystujemy różne funkcje z biblioteki LOFI,
- deklarujemy własne zmienne,
- *definiujemy własne funkcje,
- piszemy proste programy i wgrywamy ja na płytkę Arduino,
- podłączamy do płytki urządzenia wejścia (potencjometr, czujnik odległości, czujnik natężenia światła) i wyjścia (diody, buzzer).

Na zakończenie opowiadamy uczniom, co będziemy robić i czego się nauczymy podczas kolejnej lekcji: będziemy programować serwomotor – specjalny rodzaj silnika używany przy różnego rodzaju siłownikach, np. drzwiach, ramionach robota itp.