32 Pierwszy program w Arduino C

Przedmiot: Informatyka

Etap: klasa VII-VIII

Czas na realizację: 45min.

Autor: Grzegorz Troszyński

Redakcja: Joanna Skalska

Krótki opis zajęć:

Przekształcamy kod programu blokowego na tekstowy język Arduino C. Poznajemy funkcje edytora tekstowego wbudowanego w środowisko mBlock 5. Piszemy pierwszy program w Arduino C.

Cele szczegółowe:

w wyniku przeprowadzonej lekcji, uczeń:

- rozpoznaje polecenia języka Arduino C odpowiadające różnym bloczkom
- wprowadza kod programu Arduino C korzystając z podpowiedzi edytora
- tworzy w edytorze podstawową strukturę programu Arduino C
- wprowadza komentarze do programu

Środki dydaktyczne i materiały pomocnicze:

- roboty mBot
- komputery z zainstalowanym oprogramowaniem mBlock 5
- adaptery Bluetooth dongle lub kable USB dla każdego robota
- projektor z ekranem lub tablica interaktywna

Formy/metody pracy:

metoda praktyczna – pokaz z objaśnieniem
metoda problemowa – wykład konwersatoryjny
metoda praktyczna – ćwiczenia

Zgodność z podstawą programową przedmiotu informatyka:

klasy VII–VIII

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;

Przebieg zajęć:

Wprowadzanie w tematykę zajęć (10 min.)

Przeprowadź dyskusję na temat języków programowania. Zapytaj uczniów o znane im nazwy języków programowania i ich zastosowania (np. PHP – do tworzenia dynamicznych stron internetowych, programy wykonywane na serwerze, JavaScript – do stron internetowych programy wykonywane w przeglądarce, C++, Python, Java – uniwersalne języki programowania).

Zilustruj dyskusję przykładami kodu w różnych językach.

Kod maszynowy to postać programu komputerowego (zwana postacią wykonywalną lub binarną) przeznaczona do bezpośredniego wykonania przez procesor
[przykład]

Języki niskiego poziomu (assemblery) – rozkazy procesora przedstawiane są w formie literowych symboli, kod jest znacznie bardziej czytelny dla człowieka ale nadal polecenia programu odpowiadają rozkazom konkretnego procesora. Używane do programowania mikrokontrolerów, kiedy zależy nam na jak najmniejszych rozmiarach programu
[przykład]

Języki wysokiego poziomu – składnia i słowa kluczowe ułatwiają rozumienie kodu przez człowieka, są niezależne od sprzętu (procesora) na którym program będzie wykonywany.
[przykład]

Jaki jest najlepszy język programowania? Taki który najbardziej pasuje do konkretnego celu i oczywiście ten który programista zna najlepiej ;)

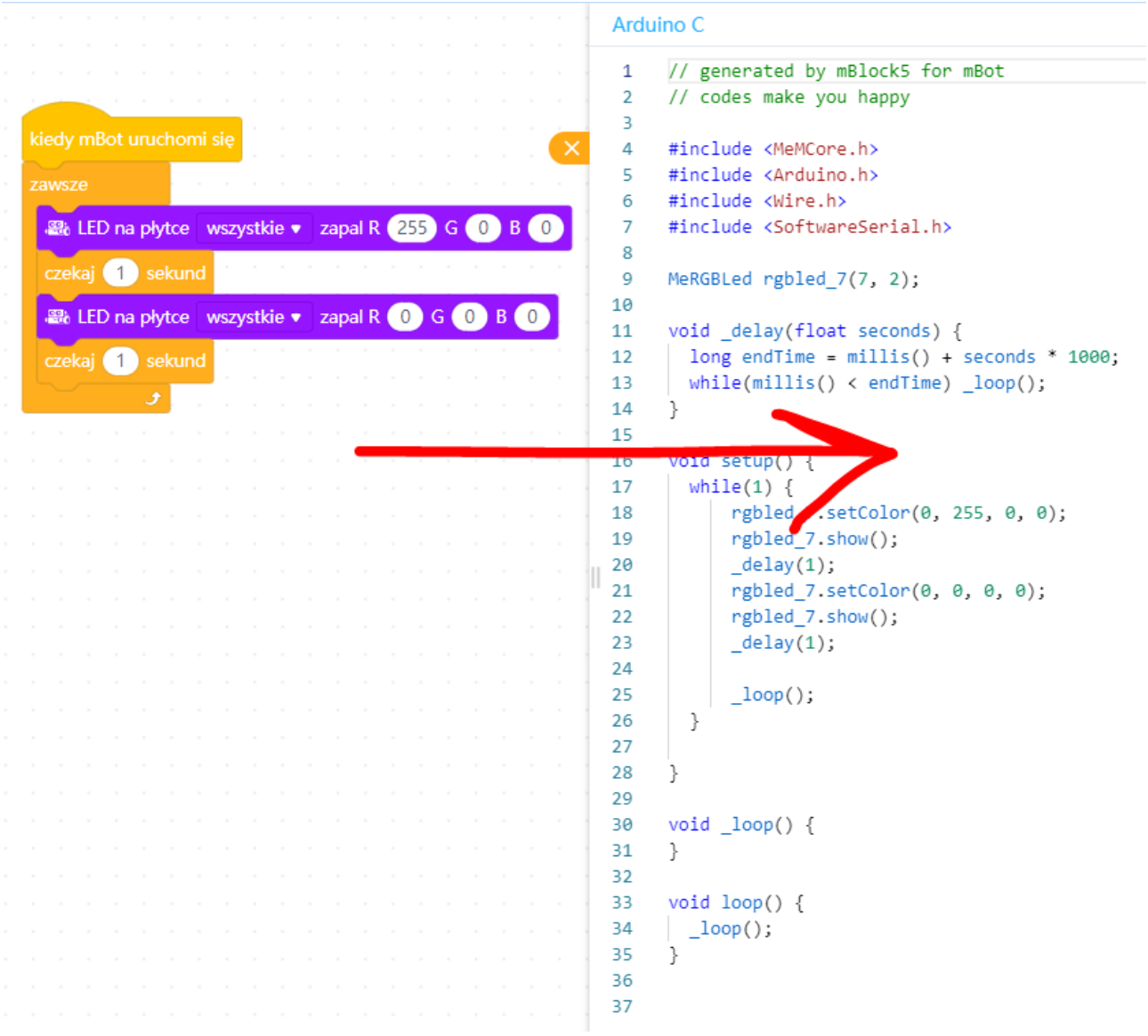
Uczniowie uruchamiają środowisko mBlock na komputerach, włączają roboty i nawiązują bezprzewodowe lub przewodowe połączenie.

Automatyczne generowanie kodu Arduino C w mBlock.

Ćwiczenie 1

Znasz już zapewne graficzne środowisko mBlock. Spróbuj ułożyć z bloczków program który będzie naprzemiennie zapalał i gasił wbudowane na sterowniku robota diody LED RGB.
Podłącz robota i załaduj ten program do jego pamięci. Działa? Doskonale.

Teraz kliknij symbol < / > w prawym górnym rogu okna programu (dostępny tylko w trybie wysyłania).



Zobaczysz kod Twojego programu, zapisany w języku tekstowym. Środowisko mBlock przetłumaczyło automatycznie Twoje bloczki na kod Arduino C. Kiedy klikasz „wyślij”, mBlock dokonuje kompliacji tego właśnie kodu na kod maszynowy, zrozumiały dla procesora Twojego robota i dopiero tak przetworzony kod ładowany jest do pamięci mBota. Tajemnicze komunikaty

które widzisz podczas ładowania pochodzą właśnie od narzędzi odpowiedzialnych za cały ten proces.

Ćwiczenie 2

Spróbuj teraz dokonać drobnych zmian w Twoim programie blokowym. Obserwuj cały czas okno z kodem Arduino C. Zmiany które robisz w bloczkach od razu będą widoczne w kodzie Arduino. Zmień np. kolor świecenia LEDów. Zamrugaj tylko prawym. Ustaw inny czas zapalania i gaszenia LEDów. Łatwo się zorientujesz, które polecenia i ich parametry odpowiadają znanym Ci bloczkom.

Edycja kodu Arduino C w mBlock IDE.

W poprzednich ćwiczeniach używaliśmy automatycznego generatora kodu. Niestety nie można w nim bezpośrednio edytować kodu Arduino. Poza tym, automatycznie generowany kod zawiera wiele elementów nadmiarowych – dodanych po to by zapewnić jego poprawne działanie w każdych warunkach i umożliwić dalszą rozbudowę. Początkującym programistom może to jednak sprawiać problemy ze zrozumieniem kodu.

mBlock IDE zawiera również edytor, pozwalający na ręczne wprowadzanie kodu.

Ćwiczenie 3

Kliknij zakładkę Arduino C. Otworzysz w ten sposób okno edytora. Spróbuj teraz wpisać kod z poniższego rysunku:

```
1  // Mój własny program w Arduino C
2
3  #include <MeMCore.h>
4
5  MeRGBLed rgbled_7(7, 2);
6
7  void setup() {
8
9  }
10
11 void loop() {
12   rgbled_7.setColor(0, 0, 0, 255);
13   rgbled_7.show();
14   delay(1000);
15   rgbled_7.setColor(0, 0, 0, 0);
16   rgbled_7.show();
17   delay(1000);
18 }
19
```

To jest ten sam program, który tworzyliśmy w ćwiczeniu 1, ale tym razem wszystkie polecenia wpisaliśmy samodzielnie (i mruga na niebiesko a tamten na czerwono). Kiedy klikniesz "wyślij" ten program zostanie skompilowany i załadowany do pamięci robota. Jeżeli w pliku projektu masz też program zbudowany z bloczków i chcesz załadować go na robota, przełącz się na zakładkę "Bloki".

Komentarze

Łatwo zauważyć, że edytor automatycznie koloruje wprowadzany kod. Pomaga to odróżnić poszczególne rodzaje poleceń, ich parametry i komentarze.

```
28  // Komentarze sa bardzo pomocne - pisząc program uzywaj ich jak najwięcej
29  // Dzięki nim inni łatwiej zrozumieją Twoje genialne programy ;)
30  // A Ty łatwiej sobie przypomnisz o co Ci chodziło kiedy je pisałeś :))
31  // Komentarz na jedną linię. Zaczyna się od // i jest komentarzem do końca tej linii
32
33  int a       // ta linia będzie częścią programu (oprócz komentarza oczywiście)
34
35  // int b
36  // ale jeśli wpiszesz symbol komentarza na początku linii to cała linia bedzie komentarzem
37  /* A to jest komentarz który ciągnie się
38  przez kilka
39  linii
40  aż do znaku */
```


```
28  // Komentarze sa bardzo pomocne - pisząc program uzywaj ich jak najwięcej
29  // Dzięki nim inni łatwiej zrozumieją Twoje genialne programy ;)
30  // A Ty łatwiej sobie przypomnisz o co Ci chodziło kiedy je pisałeś :))
31  // Komentarz na jedną linię. Zaczyna się od // i jest komentarzem do końca tej linii
32
33  int a       // ta linia będzie częścią programu (oprócz komentarza oczywiście)
34
35  // int b
36  // ale jeśli wpiszesz symbol komentarza na początku linii to cała linia bedzie komentarzem
37  /* A to jest komentarz który ciągnie się
38  przez kilka
39  linii
40  aż do znaku */
```

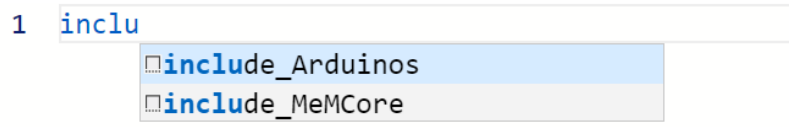

Ćwiczenie 4

Dodaj komentarze do programu z poprzedniego ćwiczenia. Postaraj się wyjaśnić sobie i innym co robią poszczególne fragmenty programu. Jeśli jeszcze nie znasz wszystkich poleceń – nie szkodzi :)

```
// Mój własny program w Arduino C

// Dołączamy do programu wszystkie zasoby potrzebne
// do działania sterownika mCore. Znajdują się one
// w bibliotece zainstalowanej razem z programem mBlock

#include <MeMCore.h>

// Tworzymy obiekt klasy MeRGBLed o nazwie rgbled_7 (możesz wymyślić inną)
// To są nasze dwie diody LED na płytce
// Brzmi tajemniczo ale na razie się tym nie przejmujemy

MeRGBLed rgbled_7(7, 2);

// Funkcja setup() musi byś zdefiniowana w programie Arduino
// Wykonuje się tylko raz przy starcie programu
// W tym przypadku jest pusta - nic nie robi ale musi być :)

void setup() {

}

// Funkcja loop() to drugi obowiązkowy element programu w Arduino
// Jest to nieskończona pętla, właściwy kod programu umieszczasz w tej pętli.

void loop() {
  rgbled_7.setColor(0, 0, 0, 255); // ustawiamy kolor obu LEDów jako czerwony
  rgbled_7.show(); // zapalamy LEDy na ustawiony wcześniej kolor
  delay(1000); // czekamy 1000mS czyli 1s
  rgbled_7.setColor(0, 0, 0, 0); // ustawiamy kolor obu LEDów jako czarny (brak koloru)
  rgbled_7.show(); // zapalamy LEDy na ustawiony wcześniej kolor czyli w tym wypadku gasimy je
  delay(1000); //czekamy 1000mS
}

```

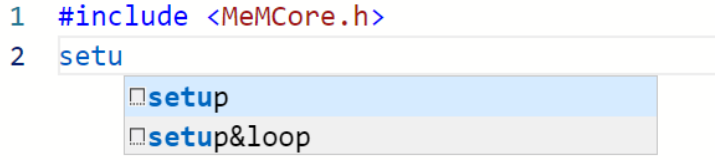
Podpowiedzi edytora

Zauważ, że gdy zaczniesz pisać w edytorze mBlock, program automatycznie podpowiada Ci znane mu słowa kluczowe. Na przykład, zaczynamy pisać pierwszą linie programu z poprzednich ćwiczeń:



Wybierz z listy MeMCore i zobacz efekt:

Podobnie jednym kliknięciem możesz stworzyć szkielet programu Arduino zawierający funkcje setup() i loop()



Edytor będzie Ci pomagał podpowiadając właściwą składnię poleceń i ich parametry.

Dokumentację biblioteki i kody przykładowych programów ilustrujących użycie różnych czujników i modułów Makeblock znajdziesz tutaj:
http://learn.makeblock.com/cn/Makeblock-library-for-Arduino/index.html

Używając przykładowych programów pamiętaj by zmienić plik nagłówka na odpowiedni dla Twojego robota. Dla robota mBot zamiast #include <MeOrion.h> użyj #include <MeMCore.h>

Podsumowanie i ewaluacja (5min.)

Nauczyciel zadaje uczniom pytania:

- Co najbardziej podobało się Wam podczas dzisiejszej lekcji?
- Z czym mieliście największe problemy?
- Czego nauczyliście się na dzisiejszej lekcji?
- Do czego można wykorzystać umiejętności zdobyte na tej lekcji?