MoboLab – roboty i tablety w Twojej szkole
Obszar II. „Stwórz własnego robota”
Scenariusze lekcji i zajęć pozalekcyjnych

SCENARIUSZ 2. STERUJEMY DIODĄ LED

scenariusz zajęć pozalekcyjnych

autor: Wojciech Karcz, Michał Podziomek

redakcja: Agnieszka Koszowska

SŁOWA KLUCZOWE:

Arduino, programowanie, mikrokontroler, dioda LED, Arduino IDE

KRÓTKI OPIS ZAJĘĆ:

Podczas zajęć uczniowie i uczennice uczą się, jak działa **dioda LED** oraz jak ją podłączyć do **Arduino**. Poznają i/lub utrwalają podstawowe pojęcia programistyczne (skrypt, program, algorytm, sterowanie, warunek, pętla). Wykonują zadanie: budują układ elektroniczny z wykorzystaniem Arduino, **płytki stykowej**, **diody LED** i **oporników**. Zadanie polega na podpięciu przez uczniów układu do nowego, wybranego przez nich pinu, przypisaniu go w programie za pomocą polecenia **pinMode(nr pinu, OUTPUT)** i sprawieniu, że dioda będzie migać (tzn. świecić się przez 2 sekundy, gasnąć na 1 sekundę i powtarzać cały cykl).

WIEDZA I UMIEJĘTNOŚCI ZDOBYTE PRZEZ UCZNIA / UCZENNICĘ:

- wie, czym są mikrokontrolery i do czego służą,
- zna pojęcia: mikrokontroler, skrypt, program, algorytm, sterowanie, warunek, pętla,
- zna projekt Arduino, wie, czym jest platforma Arduino, z jakich części się składa,
- potrafi w podstawowym stopniu samodzielnie obsługiwać Arduino (podłączyć płytkę do komputera, wgrać prosty program),
- wie, co to jest dioda LED,
- potrafi poprawnie podłączyć diodę LED do Arduino,
- zna podstawowe elementy interfejsu środowiska programistycznego Arduino

IDE i podstawowe komendy języka Arduino IDE: **pinMode()**, **digitalWrite()**, **delay()**, **digitalWrite(),**
- zna podstawowe elementy języka **Scratch**, potrafi stworzyć prosty skrypt w tym języku.

GRUPA DOCELOWA:

Starsze klasy szkoły podstawowej (VII-) i klasy gimnazjalne (po dostosowaniu: możliwość realizacji w młodszych klasach: I-III i IV-VI szkoły podstawowej). W młodszych klasach – możliwość wykorzystania programu mBlock (po przejściu scenariusza nr 18. *Programowanie Arduino z wykorzystaniem programu mBlock*) lub Scratch for Arduino (po przejściu scenariusza nr 1. *Wprowadzenie do Arduino*).

LICZBA UCZNIÓW/UCZENNIC W GRUPIE:

Liczba optymalna: 12, liczba maksymalna: 16

CZAS TRWANIA ZAJĘĆ:

90 min (lub 2 x 45 minut)

STOPIEŃ TRUDNOŚCI/SKOMPLIKOWANIA (w skali od 1 do 5 dla obszaru II. „Stwórz własnego robota”):

1

POTRZEBNY SPRZĘT I OPROGRAMOWANIE:

- komputer (przenośny lub stacjonarny),
- program Arduino IDE (do pobrania ze strony: http://www.arduino.org/downloads),
- (opcjonalnie) program mBlock (do pobrania ze strony: http://www.mblock.cc/download/) lub Scratch for Arduino (do pobrania ze strony: http://s4a.cat/),
- płytka Arduino UNO i kabel USB A-B (dla każdego uczestnika lub dla pary uczestników),
- płytka stykowa,
- oporniki o różnych wartościach (np. 220 omów, 330 omów, 1,2K omów, 2,2 omów),
- przewody połączeniowe,
- diody LED w różnych kolorach,
- projektor i laptop (w części teoretycznej).

CO NALEŻY PRZYGOTOWAĆ PRZED ZAJĘCIAMI:

- zainstalować program Arduino IDE,
- (opcjonalnie): zainstalować program **mBlock** lub **Scratch for Arduino**,
- sprawdzić, czy wszystkie komputery wykrywają podłączone Arduino,
- przeczytać dokładnie scenariusz,
- zapoznać się z materiałami dodatkowymi (w części „Pigułka wiedzy i inspiracji"),
- wykonać samodzielnie zadania zawarte w scenariuszu,
- przy każdym stanowisku komputerowym rozłożyć elementy zestawu Arduino, które będą wykorzystywane na tych zajęciach,
- dopasować stopień trudności zadania do potrzeb i możliwości klasy, dla której organizowane są zajęcia według wskazówek zawartych w scenariuszu.

KOMPETENCJE OSOBY PROWADZĄCEJ:

- wie, czym jest projekt Arduino, zna podstawowe informacje o projekcie,
- potrafi przynajmniej w stopniu podstawowym obsługiwać Arduino,
- zna podstawowe pojęcia z zakresu elektroniki,
- zna podstawowe pojęcia programistyczne,
- wie, dlaczego warto uczyć się programowania i jakie korzyści daje posiadanie umiejętności programistycznych,
- potrafi zachęcić do nauki programowania zarówno chłopców, jak i dziewczynki.

PRZEBIEG ZAJĘĆ:

Podłączenie Arduino, uruchomienie programu Arduino IDE i przekazanie lub przypomnienie podstawowych informacji

Uwaga! Tę część scenariusza warto realizować na każdych zajęciach (tych i kolejnych realizowanych wg scenariuszy z obszaru II. „Stwórz własnego robota") w takim zakresie, jaki jest potrzebny, do czasu aż podstawowy materiał zostanie utrwalony.

Podłączenie Arduino – 10 minut

Arduino programujemy za pomocą komputera. A więc najpierw musimy podłączyć Arduino do komputera za pomocą kabla USB A-B. W języku i środowisku programistycznym Arduino IDE tworzy się programy, które następnie wgrywa się

(przesyła za pomocą kabla) na Arduino. Kiedy program zostanie wgrany, (trwa to kilku sekund), będzie on wykonywany bez przerwy (w pętli). Teraz pokazujemy uczniom, jak podłączyć Arduino do komputera.

Po podłączeniu uruchamiamy program Arduino IDE. Sprawdzamy, czy na każdym komputerze został poprawnie wykryty port, do którego jest podłączone Arduino (trzeba kliknąć w górnym menu w „Narzędzia", a następnie w „Port"). Uczulamy uczniów na to, że czasem mogą pojawiać się problemy z wykryciem Arduino na komputerze i warto wtedy sprawdzić, czy port jest aktywny.

Następnie uruchamiamy program Arduino IDE. Na platformie Windows należy wyszukać program w menu Start. Na platformie Linux Ubuntu należy odnaleźć program w menu Ubuntu w lewym górnym rogu ekranu.

Uwaga! Osoba prowadząca powinna co najmniej dzień wcześniej sprawdzić, czy program Arduino IDE jest zainstalowany na wszystkich komputerach, z których będą korzystali uczniowie.

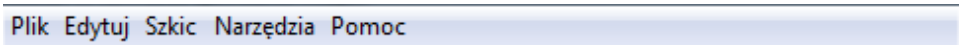
Omawiamy interfejs programu (w razie potrzeby przypomnienia) – ok. 10 minut

Warto korzystać z anglojęzycznego interfejsu programu, ponieważ jest to język, w którym będziemy pisać programy. Języki programowania w zdecydowanej większości są anglojęzyczne i zazwyczaj nie są tłumaczone na inne języki. Język angielski jest podstawowym językiem programistów, więc zarówno uczniowie, jak i nauczyciele powinni zapoznać się z podstawową anglojęzyczną terminologią dotyczącą programowania. Nie jest jej wiele, a z pomocą przychodzą słowniki online i polskojęzyczne materiały edukacyjne dostępne w internecie. Umiejętność wyszukiwania informacji i rozwiązań napotkanych problemów jest podstawową cechą dobrego programisty.

Jeżeli zaistnieje taka potrzeba, mamy możliwość zmiany języka programu. Wykonujemy ją w następujący sposób: klikamy **File**, następnie **Preferences** i w zakładce **Settings** wybieramy z listy **Editor Language**. Dla polskiej wersji językowej będzie to: Plik > Preferencje > Ustawienia > Język edytora. Po dokonaniu zmiany należy zamknąć i uruchomić ponownie program. Zmiana pozwoli nam porównać komunikaty wyświetlane w wersji polskiej i angielskiej, a także przełączać język w zależności od potrzeb.

Po uruchomieniu programu powinniśmy zobaczyć białe okno z tekstem, niebieskozielonymi elementami graficznymi oraz czarnym grubym paskiem w dole okna. U góry okna widzimy następujące menu:

lub



File (Plik): tu możemy tworzyć nowe projekty (New), zapisywać je (Save, Save As), otwierać (Open, Open Recent) oraz przeglądać przykłady gotowych programów (Examples).
Edit (Edytuj): menu edycji, pozwalające na kopiowanie i wklejanie tekstu, oraz inne metody jego modyfikowania.
Sketch (Szkic): menu pozwalające na skompilowanie kodu, czyli przetłumaczenie go na język zrozumiały dla komputera oraz załadowanie go na Arduino.
Tools (Narzędzia): menu narzędzi, tu m.in. możemy wybrać rodzaj płytki Arduino, którą posiadamy, oraz port USB, do którego płytka jest podłączona do naszego komputera.
Help (Pomoc): menu pomocy.

Poniżej menu widzimy następujące przyciski:

Pierwszy: **Verify** (Zweryfikuj) pozwala na weryfikację poprawności kodu. Program Arduino sprawdzi, czy kod, który napisaliśmy, jest poprawny, to znaczy, czy może zostać skompilowany (przetłumaczony na język komputera).
Jeżeli wybierzemy z menu polecenia: File > Examples > 01.Basic > Blink (Plik > Przykłady > 01.Basics > Blink) i klikniemy ten przycisk, w czarnym polu u dołu okna powinny się wyświetlić różne informacje napisane białą czcionką. Świadczy to o tym, że program może być skompilowany i wysłany na płytkę Arduino.

Drugi: **Upload** (Wgraj) działa podobnie do Verify, z tą różnicą, że po udanej kompilacji od razu wysyła kod na płytkę Arduino, jeżeli ta jest poprawnie podłączona do naszego komputera.

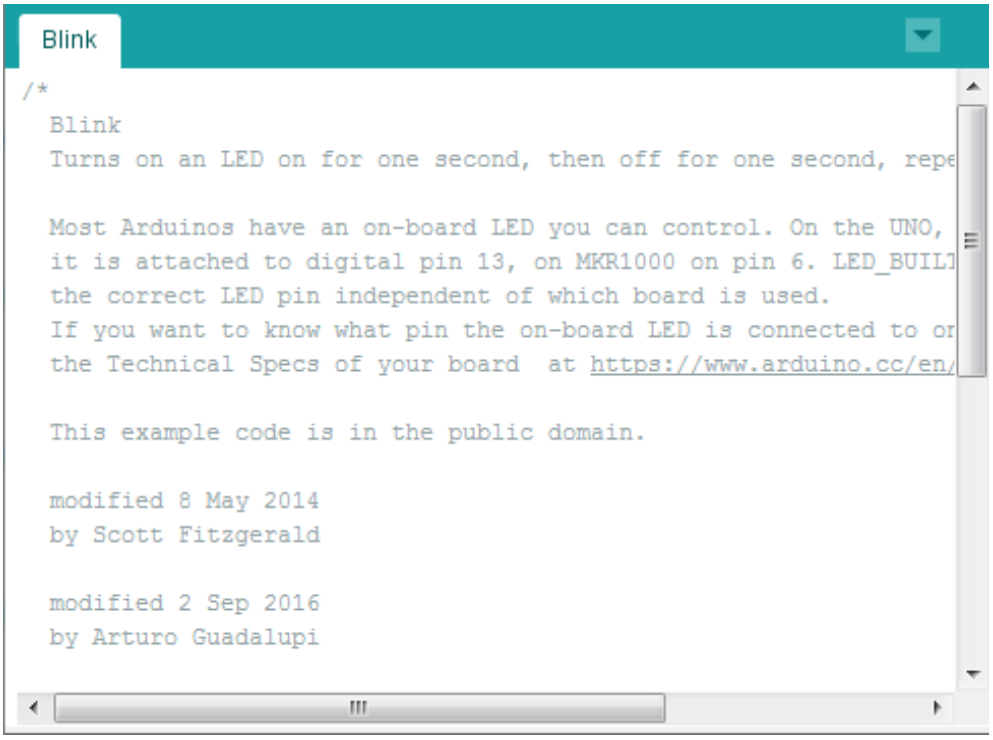
Trzeci: **New** (Nowy) otwiera nowy pusty projekt.

Czwarty: **Open** (Otwórz) otwiera wcześniej zapisany projekt.

Piąty: **Save** (Zapisz) zapisuje projekt.

Szósty: znajdujący się na samym końcu po prawej stronie otwiera **Serial Monitor** (Monitor Portu Szeregowego). Jest to okno, w którym widzimy, jakie sygnały i informacje płytka Arduino wysyła do naszego komputera, i pozwala nam na wysyłanie sygnałów i informacji do płytki. Żeby móc wysyłać i odbierać sygnały, musimy płytkę w tym celu najpierw zaprogramować.

Poniżej znajduje się okno z naszym programem. Programy w Arduino nazywamy „szkicami" (sketches).

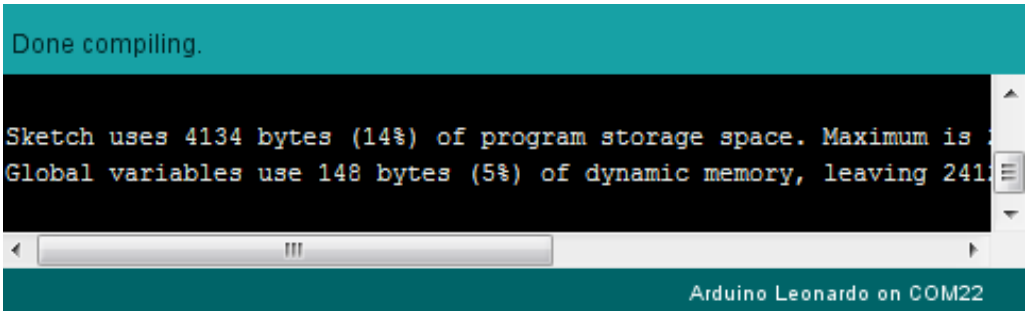


Po lewej stronie znajduje się nasza jedyna zakładka. W tym przypadku jest to szkic załadowany wcześniej z menu Examples o nazwie **Blink** (mruganie diodą wbudowaną w moduł Arduino).

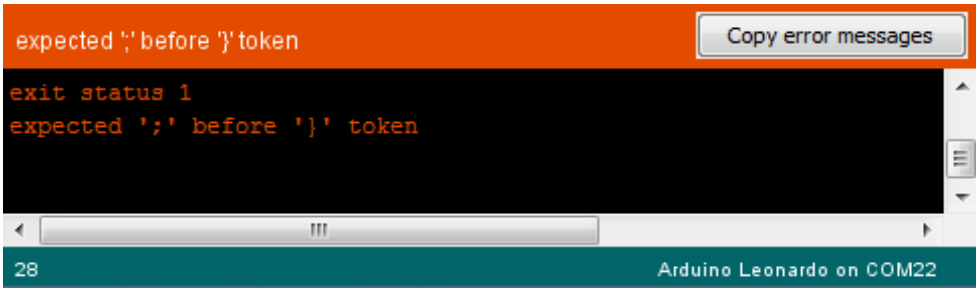
Po prawej stronie znajduje się rozwijana ikonka pozwalająca na nawigację po zakładkach programu.

Poniżej mamy kod – tekst naszego szkicu. Z jego pomocą będziemy tworzyć programy, które następnie uruchomimy na płytkach Arduino podłączonych do komputera. Poniżej znaduje się okno, w którym program Arduino komunikuje się z nami, pokazując błędy podczas kompilacji programu lub informując o sukcesie. Zwykle okno na początku wygląda następująco:

Po udanej kompilacji okno będzie wyglądać tak, jak na poniższym rysunku, i informować nas, ile pamięci na płytce Arduino zabiera nasz program:



Jeżeli podczas kompilacji wystąpił błąd, w oknie pokaże się komunikat w kolorze pomarańczowym, informując nas, jaki błąd popełniliśmy. Na poniższym przykładzie brakuje średnika na końcu linii:



Piszemy program na Arduino (w razie potrzeby przypomnienia) przypomnienia – ok. 10 minut

Ten scenariusz zawiera instrukcję, w jaki sposób napisać program na Arduino. Korzystając z poniższych wskazówek omawiamy strukturę programu.

Tworzymy nowy projekt. Z menu **File** wybieramy polecenie **New**, albo klikamy w przycisk **New** w interfejsie programu. Zobaczymy wówczas nowe okno z następującym tekstem:

```
void setup() {
  // put your setup code here, to run once:
}
void loop() {
  // put your main code here, to run repeatedly:
}
```

Jest to podstawowa struktura programu Arduino. Zgodnie z opisem w języku angielskim, część „setup" jest wykonywana tylko raz, na początku uruchomienia programu, po czym wykonywana jest część „loop" na okrągło, czyli w tzw. **pętli**. Jeżeli uczestnicy zajęć brali udział w lekcjach lub zajęciach, na których uczyli się Scratcha, termin „pętla" powinien być im znany. Tekst, który następuje po znakach **//** to komentarze – tu możemy wpisać to, co chcemy. Zazwyczaj komentarze wpisywane są jako informacja dla programisty, co wykonuje dana część programu. Komentarz musi być w tej samej linii co znaki **//**.

Piszemy program który będzie nam wyświetlał informację w **Serial Monitor**. Robimy to wpisując w okno następujący kod - wyjaśnienia w komentarzach:

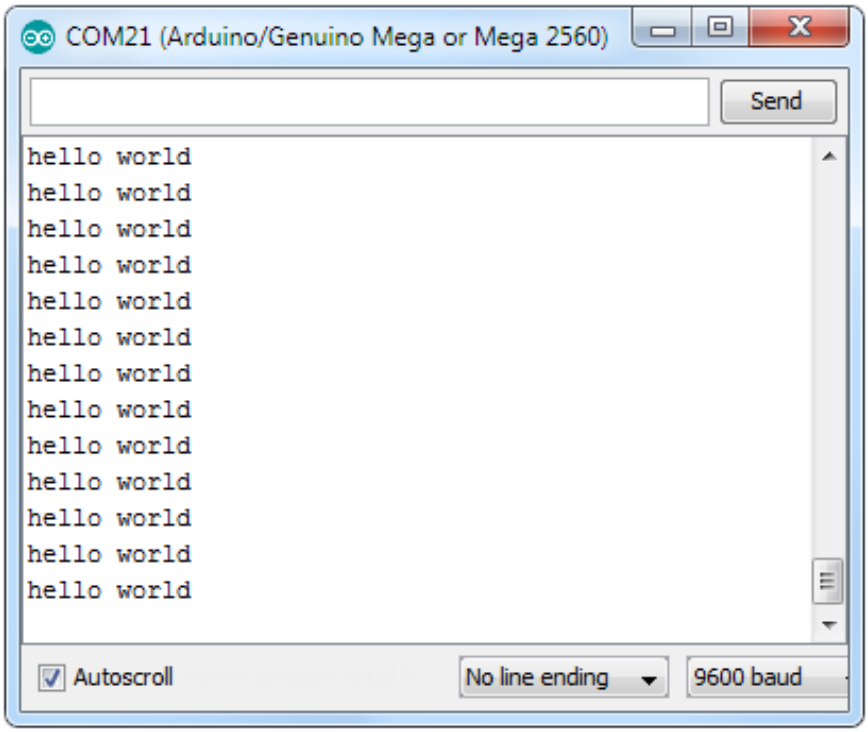
```
void setup() {
  // mówimy Arduino że będzie się komunikowało z komputerem za pomocą przewodu USB
 // skrót USB oznacza Universal Serial Bus - stąd nazwa komendy „Serial":
  Serial.begin(9600);
}
void loop() {
   // wysyłamy tekst „hello world", czyli „witaj świecie" z Arduino do komputera
```

```
  Serial.println("hello world");

   // czekamy 1000 milisekund, czyli jedną sekundę
  delay (1000);
}
```

Następnie klikamy przycisk **Upload**. Jeżeli wystąpiły błędy, należy je poprawić - program powinien być przepisany bardzo starannie i bez błędów. Potem klikamy w przycisk **Serial Monitor** lub wybieramy z menu **Tools>Serial Monitor** (Narzędzia> Monitor Portu Szeregowego).

Powinniśmy zobaczyć okno w którym co sekundę pojawia się napis „hello world”:



Skąd wiedzieliśmy, co wpisać, by uzyskać żądany efekt? Otóż każdy język programowania zawiera słowa o konkretnym znaczeniu. Lista słów dostępnych w Arduino jest dostępna pod poniższym adresem:
https://www.arduino.cc/en/Reference/HomePage

O wiele wygodniej na początek jest korzystać z przykładów programów dołączonych do Arduino **(File>Examples) (Plik>Przykłady)**. Języka programowania uczymy się w bardzo podobny sposób, jak języka obcego - musimy poznać jego **słowa**, **składnię**, **gramatykę**, **ortografię**.

I tak, bardzo upraszczając: **„Serial”** jest to **funkcja**. Ma ona swoje podfunkcje, które potrafią wykonywać różne operacje, np. **„println” drukuje linię**, a **„begin” rozpoczyna komunikację z komputerem**. Te podfunkcje mają swoje **parametry**, które wpisujemy w nawiasie. „println” musi wiedzeć, co konkretnie ma wydrukować („hello world”). „begin” musi wiedzieć, z jaką częstotliwością ma rozmawiać z komputerem (w tym wypadku - 9600 sygnałów na sekundę).

Pętla programu działa bardzo szybko. Na przykład procesor płytki Arduino Mega ma częstotliwość pracy 16 MHz. **Hz**, czyli Hertz, to jednostka **częstotliwości** – określa jeden **cykl** (czyli jedno zdarzenie) na sekundę. *MHz* to MegaHertz – czyli jeden milion Hertzów. Tak więc przykładowy moduł Arduino Mega może pracować z prędkością 16 milionów zdarzeń na sekundę.

Zadanie dla uczniów:

Zmodyfikujcie wiadomość, którą Arduino wysyła do komputera, i załadujcie kod ponownie na płytkę. Czy Wasza wiadomość musi być w cudzysłowie, żeby program zadziałał?

Uwaga! Część dotycząca rozpoczynania pracy z Arduino występuje na wszystkich zajęciach. Zalecamy powtarzanie jej (w niezbędnym zakresie) na każdych zajęciach, do momentu w którym uczniowie będą w stanie samodzielnie wykonać podstawowe zadania. W przypadku grup o niskiej rotacji uczestników, część ta będzie zajmowała coraz mniej czasu za każdym powtórzeniem. Może też być potrzebna praca indywidualna z osobami, które nie będą nadążać za grupą.

Poznajemy diodę LED (z wykorzystaniem diod, rezystorów i płytki stykowej). Cz. 1 – 15 minut

Zajęcia są rozwinięciem pierwszych zajęć wprowadzających (Scenariusz 1. *Wprowadzenie do Arduino*). Podczas pierwszych zajęć uczniowie poznawali podstawy obsługi Arduino oraz uczyli się posługiwać wbudowaną diodą LED przy pinie 13. Podczas tych zajęć skupimy się na podłączeniu do Arduino dwóch podstawowych elementów elektronicznych, jakimi są diody LED oraz rezystory (oporniki).

Zajęcia można zacząć od pytań nawiązujących do materiału z zajęć wprowadzających, np.:

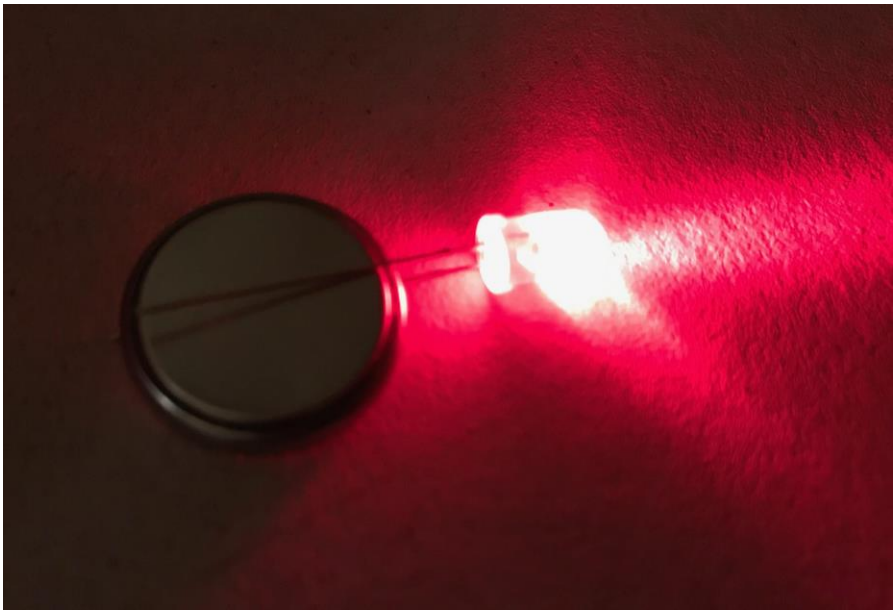
- *Co robiliśmy podczas zajęć?*
- *Co to jest Arduino?*
- *W jaki sposób kontrolowaliśmy wbudowaną diodę?*
- *Co trzeba zrobić, żeby zaprogramować Arduino?*

Podczas tych zajęć skupimy się bardziej na stronie elektronicznej niż programistycznej, ponieważ Arduino to nie tylko pisanie programów w języku Arduino IDE, ale też konstruowanie układów elektronicznych. Dobrze, jeżeli uczniowie mają już podstawową wiedzę na temat elektroniki (warto to rozpoznać na początku zajęć). Jeżeli nie, warto w tym momencie wprowadzić i krótko omówić takie pojęcia, jak: *prąd elektryczny, obwód elektryczny, źródło prądu, przewodnik*. Więcej informacji na ten temat można znaleźć w materiałach dodatkowych.

Po omówieniu podstawowych pojęć dotyczących elektroniki lub jeśli wiemy, że poziom wiedzy uczniów jest wystarczający, rozpoczynamy dyskusję na temat źródeł światła. Możemy zadać takie pytania jak:

- *Jakie mogą być źródła światła?*
- *Jakie typy oświetlenia stosujemy w różnych miejscach (w domu,* w samochodach*, w sklepach, na ulicach)?*

Powinna tutaj paść odpowiedź, że jednym ze źródeł światła mogą być diody LED stosowane np. w pilotach do telewizora, latarkach itp. W tym momencie pokazujemy diody dostępne w zestawach Arduino i dajemy je uczniom do ręki.
Dodatkowo możemy dać uczniom płaskie baterie 3V (np. CR2032) i umieścić je pomiędzy nóżkami diody, co sprawi, że dioda LED będzie świecić.

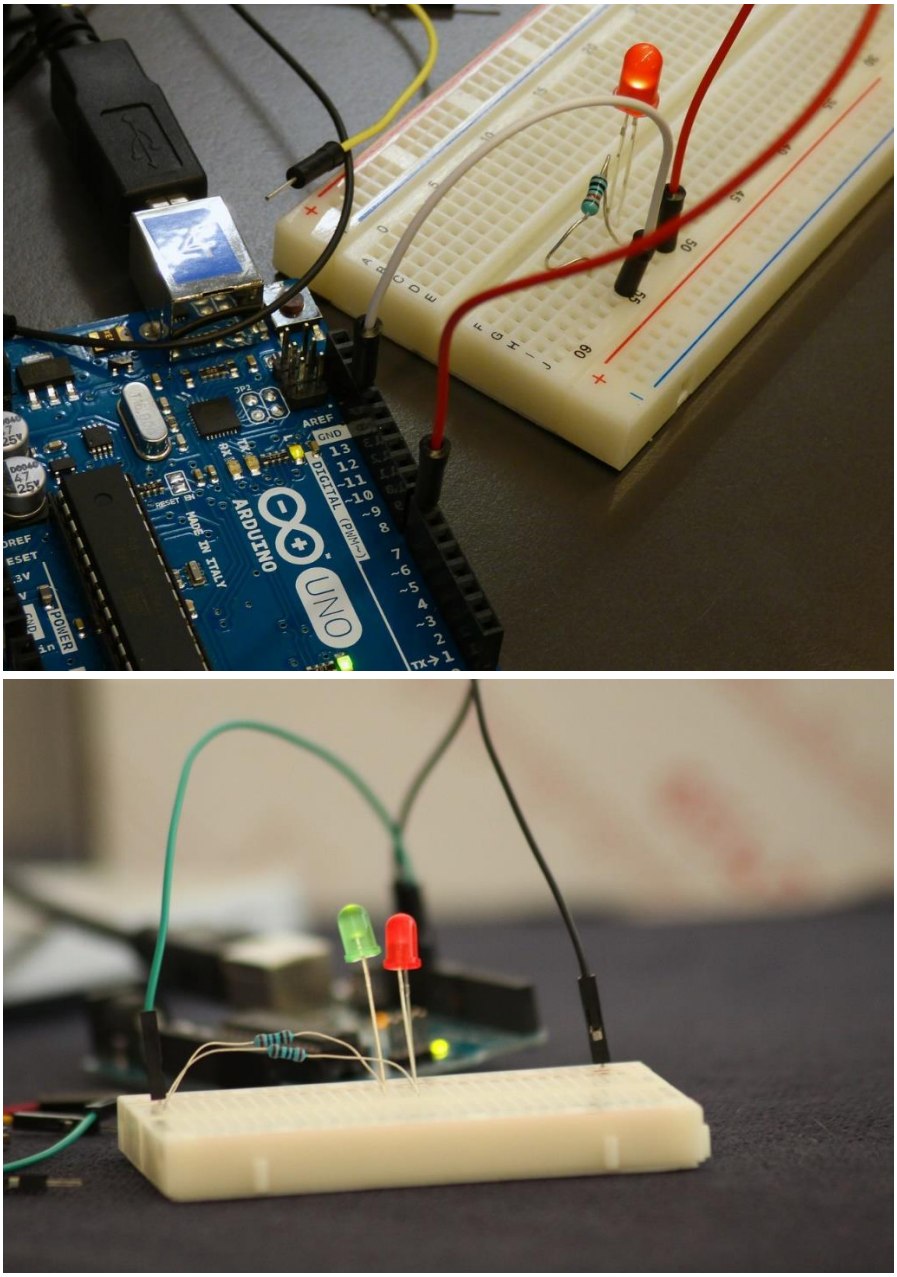


Źródło: materiały własne

Warto w tym momencie zadać uczniom pytanie: czy dioda będzie świecić tak samo, jeżeli odwrotnie przyłożymy nóżki?

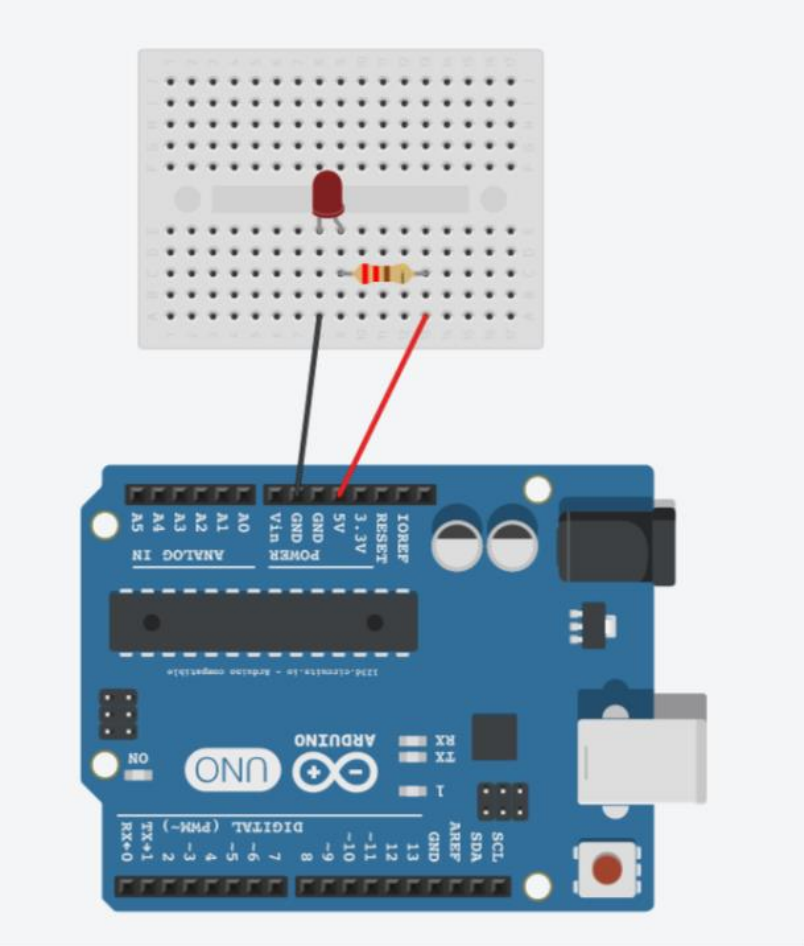
Następnie wyjaśniamy, jak działa dioda LED i gdzie ją stosujemy. Warto zaznaczyć, że dioda LED ma nóżki o różnej długości: dłuższa jest anodą, a krótsza katodą. Plus zawsze podpinamy do anody (dłuższej nóżki), inaczej dioda nie będzie świecić. Więcej informacji znajduje się w materiałach dodatkowych.

W tej części zajęć skorzystamy z płytki stykowej (prototypowej) i zmontujemy prosty układ, który będziemy kontrolować za pomocą Arduino. Jednak na samym początku warto zademonstrować uczniom, w jaki sposób działa płytka stykowa i jak podłączać do niej różne elementy (np. diody, rezystory czy kabelki połączeniowe). Więcej informacji na temat płytki stykowej i łączenia prostych układów elektronicznych znajduje się w materiałach dodatkowych.



Źródło: Pixabay

UWAGA: Często uczniowie mają problem z łączeniem kabelków i drobnych elementów elektronicznych. Warto zwrócić na to szczególną uwagę i pomagać uczniom, jeśli będą mieli z tym trudności. Najczęściej układy na płytce stykowej nie działają dlatego, że ich elementy zostały nieprawidowo podłączone.



Wyjaśniamy, że dodatkowym elementem w naszym układzie jest opornik. Jednak na razie nie wyjaśniamy jego roli. Następnie wszyscy podłączają układ i sprawdzają, czy działa. Wszędzie powinna się świecić dioda LED.

W tym miejscu możliwy jest podział zajęć na dwie części (kolejna część scenariusza będzie realizowana na następnych zajęciach).

Budujemy układ z wykorzystaniem diody LED, rezystora i płytki stykowej. Cz. 2 – 20 minut

Rozpoczynamy od krótkiego przypomnienia materiału z poprzedniej części zajęć i odtworzenia układu zbudowanego na poprzednich zajęciach.

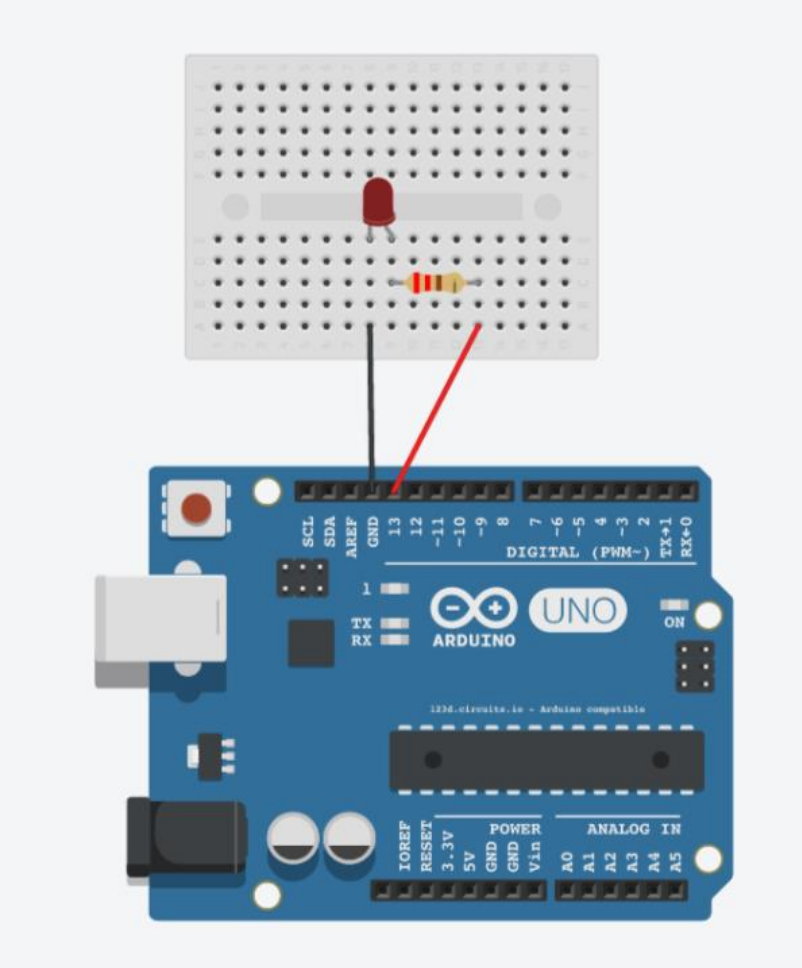
Następnie omawiamy i pokazujemy, jak działa opornik. Przygotowujemy wcześniej kilka oporników o różnych wartościach (np. 220 omów, 330 omów, 1,2K omów, 2,2 omów) i po kolei podmieniamy rezystory. Najpierw zastępujemy rezystor 220

omów rezystorem o wartości 330 omów. Potem podmieniamy rosnąco kolejne wartości.

Pytamy uczniów, czy widzą jakąś zmianę w sposobie świecenia się diody (powinna świecić coraz ciemniej). Potem wyjaśniamy, czym jest rezystor i jaką rolę pełni w elektronice. Tłumaczymy również, że bez rezystora dioda się przepali, ponieważ będzie na nią działać zbyt wysokie napięcie (ewentualnie można demonstracyjnie przepalić jedną diodę podłączoną do 5V z Arduino bez rezystora). Więcej informacji znajduje się w materiałach dodatkowych.

Sterujemy diodą LED za pomocą Arduino - 25 min

W tej części zajęć będziemy sterować już diodą LED za pomocą Arduino. Modyfikujemy nieznacznie układ, tak jak na schemacie poniżej:



Następnie wgrywamy na Arduino znany już przykładowy program Blink, który znajdziemy w zakładce Plik > Przykłady > 01.Basics > Blink. Obserwujemy, co się dzieje i pytamy o to uczniów (dioda LED powinna migać, tak samo jak dioda wbudowana w Arduino przy pinie 13).

Wyjaśniamy, że Arduino za pomocą wyjść cyfrowych steruje w tym przypadku diodą LED, ale może również sterować jakimkolwiek innym urządzeniem elektronicznym.

Odłączamy Arduino i kabel podpięty do pinu nr 13 przepinamy do jakiegokolwiek innego pinu cyfrowego (uczniowie mogą sami wybrać). Załóżmy, że będzie to pin nr 6. Następnie modyfikujemy program w odpowiednich miejscach (zamieniamy wszystkie miejsca, gdzie jest użyty pin nr 13 na 6), wgrywamy na Arduino i sprawdzamy, czy działa. W ten sposób pokazujemy uczniom, jak przypisać inne piny w Arduino IDE (nie trzeba wszystkiego podłączać do pinu nr 13).

UWAGA: Przed każdą zmianą układu na płytce stykowej, przepinaniem przewodów, dodawaniem nowych elementów elektronicznych ODŁĄCZAMY Arduino od prądu (wyjmujemy kabel USB A-B). Dzięki temu unikniemy przypadkowego zwarcia, które może zniszczyć nasze Arduino.

MOŻLIWE MODYFIKACJE DLA MŁODSZYCH KLAS:

Pracując z uczniami w młodszych klasach można wykorzystać zamiast Arduino IDE program S4A (Arduino for Scratch). W przypadku zajęć z młodszymi dziećmi warto zwrócić uwagę na ewentualne problemy z dokładnym podłączaniem przewodów.

ZADANIE SPRAWDZAJĄCE UMIEJĘTNOŚCI ZDOBYTE PODCZAS ZAJĘĆ:

Uczeń / uczennica, pracując samodzielnie albo w dwu- lub trzyosobowym zespole buduje układ z wykorzystaniem Arduino, płytki stykowej, diody LED i oporników. Zadanie polega na podpięciu przez uczniów układu do nowego wybranego przez nich pinu, przypisaniu go w programie za pomocą polecenia pinMode(nr pinu, OUTPUT) i sprawieniu, że dioda będzie migać (tzn. świecić się przez 2 sekundy, gasnąć na 1 sekundę i powtarzać cały cykl).

PIGUŁKA WIEDZY I INSPIRACJI DLA OSÓB PROWADZĄCYCH:

Kurs programowania Arduino Forbot:
http://forbot.pl/blog/artykuly/programowanie/kurs-arduino-w-robotyce-1-wstep-id936

Podstawowe informacje na temat prądu elektrycznego:
http://forbot.pl/blog/artykuly/podstawy/podstawy-elektroniki-1-napiecie-prad-opor-zasilanie-id3947

Informacja o diodach LED:
https://pl.wikipedia.org/wiki/Dioda_elektroluminescencyjna

Jak działa płytka stykowa (prototypowa):
https://pl.wikipedia.org/wiki/P%C5%82ytka_prototypowa

Scenariusz został opracowany na potrzeby projektu „MoboLab – roboty i tablety w Twojej szkole". Celem projektu jest zwiększenie kompetencji informatycznych z zakresu programowania i wykorzystywania technologii mobilnych w uczeniu się, a także kreatywności, innowacyjności i umiejętności współpracy w zespole z wykorzystaniem TIK, uczniów / uczennic z (UCZ) z 6 szkół podnadgimnazjalnych i 4 gimnazjów Wołomina i Zielonki. Projekt dofinansowany jest ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego (Regionalny Program Operacyjny Województwa Mazowieckiego na lata 2014-2020, Oś Priorytetowa X. Edukacja dla rozwoju regionu, Działanie 10.1. Edukacja ogólna i przedszkolna, Poddziałanie 10.1.2. Edukacja ogólna w ramach ZIT).