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

SCENARIUSZ 14. INSTRUMENT MUZYCZNY- GRANIE NA PRZYCISKACH

scenariusz zajęć pozalekcyjnych

autor: Michał Podziomek

redakcja: Agnieszka Koszowska

SŁOWA KLUCZOWE:

Arduino, programowanie, elektronika, buzzer, dźwięk, keyboard

KRÓTKI OPIS ZAJĘĆ:

Podczas zajęć uczniowie i uczennice poznają i/lub utrwalają wiedzę o mikrokontrolerze Arduino. Przygotowują prosty układ z wykorzystaniem **buzzera** (brzęczyka). Poznają i/lub utrwalają podstawowe pojęcia programistyczne (skrypt, program, algorytm, sterowanie, warunek, pętla, wyrażenie "if... else"). Tworzą program pozwalający grać melodie na brzęczyku z wykorzystaniem **przycisków**.

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,
- potrafi podłączyć buzzer i przełączniki do Arduino,
- zna i rozumie wyrażenie "if... else",
- zna podstawowe elementy interfejsu środowiska programistycznego Arduino IDE i podstawowe komendy języka Arduino IDE: pinMode(), digitalWrite(), delay(),

- rozumie zasadę działania funkcji digitalWrite() i potrafi wykorzystać ją w praktyce,
- 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 220 omów,
- przewody połączeniowe,
- diody LED w różnych kolorach,
- oprogramowanie mBlock (w razie potrzeby),
- brzęczyk piezoelektryczny,
- 3 przyciski,

- płytka prototypowa,
- przewody połączeniowe do płytki.
- 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,
- upewnić się że odpowiednie oprogramowanie jest zainstalowane na komputerach w miejscu przeprowadzania szkolenia - Arduino IDE, mBlock,
- 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 przypomnienie podstawowych informacji – ok. 15 minut

Uwaga! Informacje o tym, jak podłączyć Arduino, uruchomić program Arduino IDE i Scratch for Arduino, a także podstawowe informacje niezbędne przy rozpoczynaniu pracy z Arduino zawierają scenariusze 1 i 2. Tę część zajęć warto powtarzać za każdym razem w takim zakresie, jaki jest potrzebny, do czasu aż podstawowy materiał zostanie utrwalony.

Podłączamy buzzer, czyli element wydający dźwięk (brzęczyk) – 15 minut

Uwaga! Bardzo często uczniowie mają problem z łączeniem kabelków i drobnych elementów elektronicznych. Warto na to zwrócić szczególną uwagę i dużo pomagać w tym zakresie. Najczęściej układy na płytce stykowej nie działają dlatego, że po prostu przewody są źle podłączone.

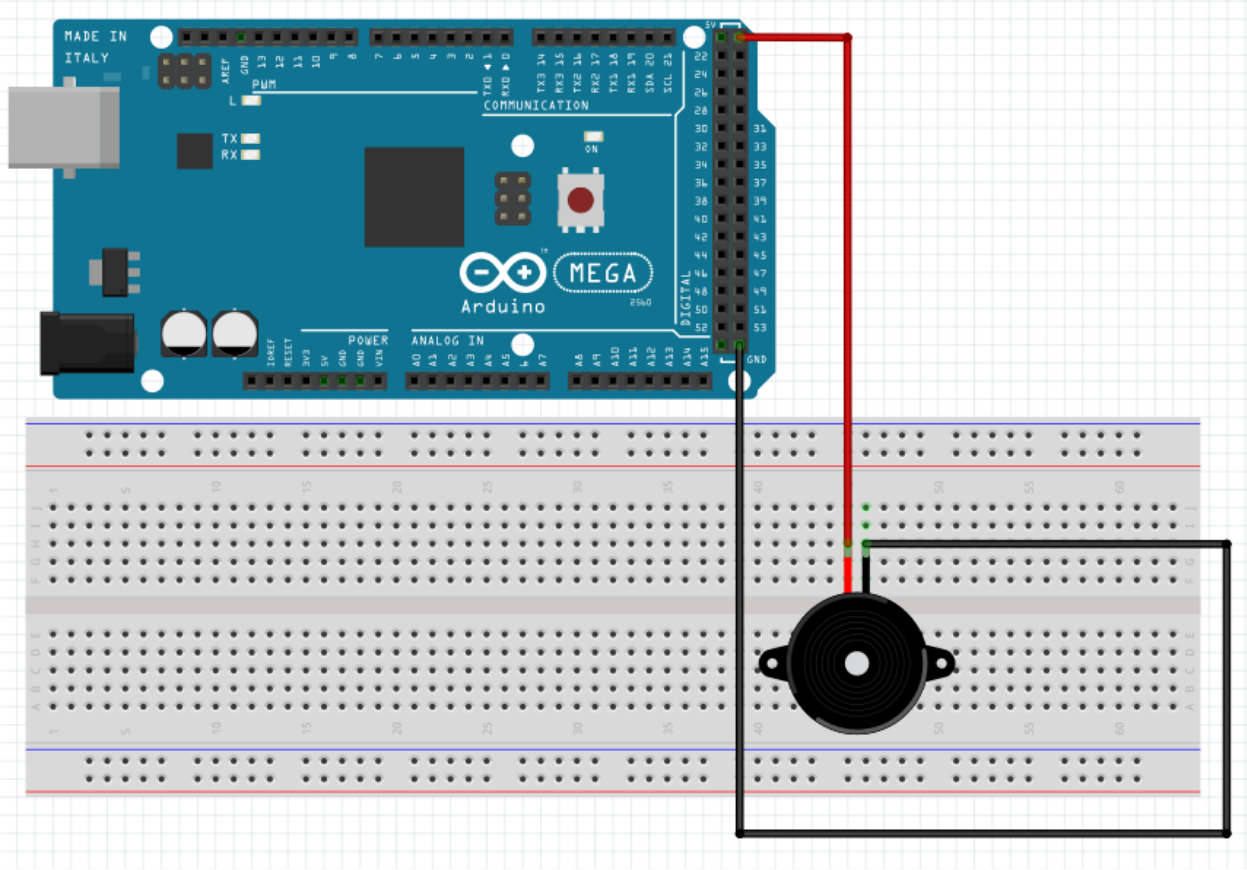
Przed każdą zmianą układu na płytce stykowej, przepinaniem przewodów, dodawaniem nowych elementów elektronicznych **ODŁĄCZAMY** Arduino od prądu **(wyjmujemy kabel USB A-B)**. Dzięki temu unikniemy przypadkowego zwarcia, które może przepalić i nawet zniszczyć Arduino.

Płytka Arduino ma **piny**. Są to rzędy otworów lub metalowych pręcików rozmieszczonych na krawędziach płytki (zależnie od modelu płytki). Dzięki tym pinom możemy wysyłać informacje do płytki Arduino, i z niej informacje odbierać. Informacje te są w postaci zero-jedynkowej: **prąd** lub **brak prądu**. Stan, w którym na pinie jest prąd nazywamy stanem **HIGH** (wysokim), a stan w kórym prądu na nim nie ma - stanem **LOW** (niskim).

Na początek szukamy pinów o oznaczeniach **PWR**, 3.3V, 5V. Są to piny, na których zawsze płynie prąd, kiedy płytka Arduino jest włączona, czyli ich stan jest HIGH. Następnie szukamy pinu **GND**. Jest to pin który jest zawsze uziemiony, i jego stan to zawsze LOW.

Jeżeli połączylibyśmy dowolny pin PWR z GND, nastąpiłoby zwarcie. Prąd zacznie uciekać z układu z ogromną prędkością. To tak jakbyśmy nagle zburzyli tamę na rzece, i woda zaczęłaby wylewać się ze zbiornika za tamą z ogromną prędkością, porywając ze sobą wszystko pod drodze. Nie chcemy tego robić, bo może to uszkodzić elementy na płytce. (w tej analogii PWR jest zbiornikiem tamy, a GND - miejscem najniżej położonym, do którego dąży ogromna fala powstała po rozbiciu tamy, niszcząc po drodze miasta i wsie - czyli układy naszego procesora i płytki).

Jeżeli natomiast podłączymy buzzer tak, by jeden pin był zasilony (PWR) a drugi uziemiony (GND), prąd przepływający przez niego zachowa się tak, jak woda spadająca na turbiny tamy wodnej - zacznie wykonywać pracę, napędzając układ. Spróbujmy tak zrobić.



Oto schemat podłączenia buzzera. Jedno z podłączeń jest zapięte na +5V (lub tzw. PWR), drugie na GND. Jeżeli podłączyliśmy układ poprawnie, buzzer powinien zacząć wydawać dźwięk. Jeżeli przerwiemy układ, czyli odepniemy jeden z przewodów, dźwięk ustanie.

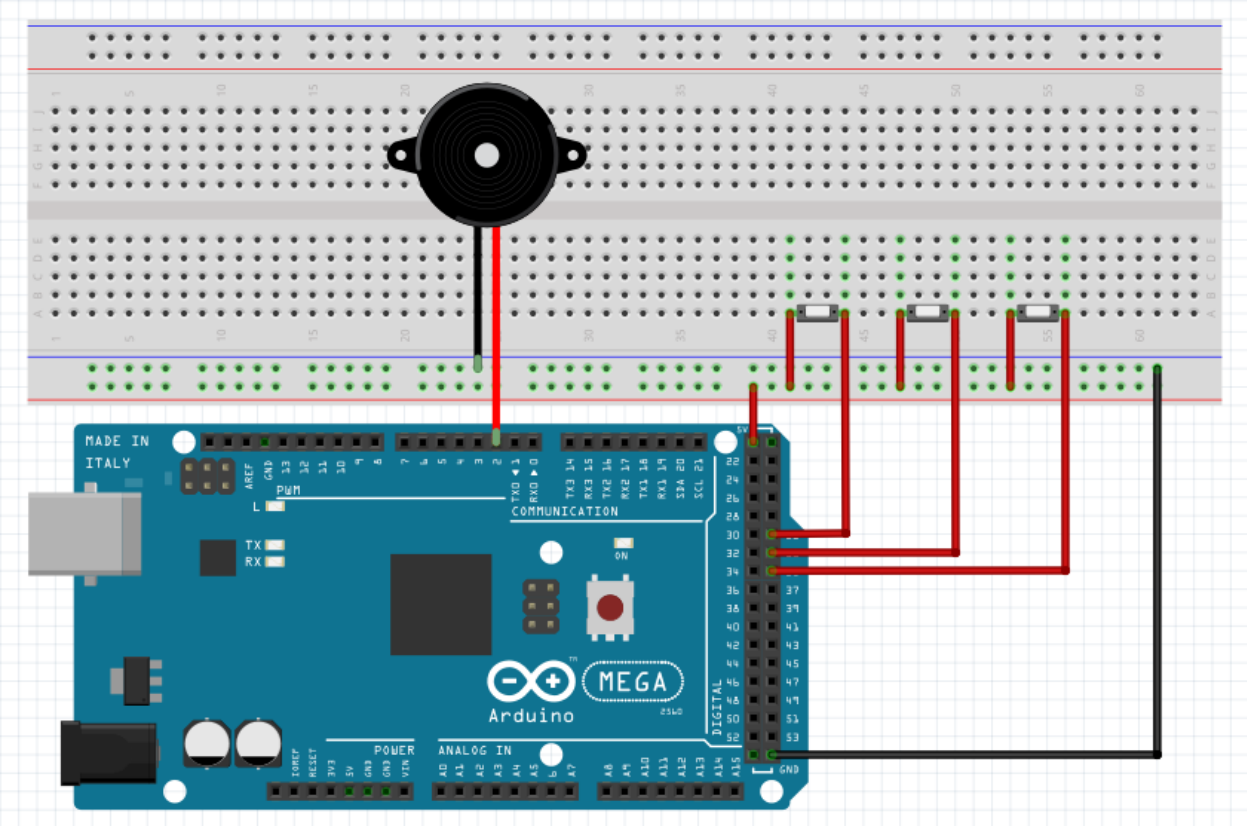
Jest to bardzo prosty układ, w którym wykorzystujemy Arduino jako źródło prądu, bez używania procesora. Możemy włączać i wyłączać buzzer stykając i rozłączając przewody.

Nasz procesor po zaprogramowaniu będzie robił dokładnie to samo co robimy my - tylko setki tysięcy razy na sekundę.

Zadanie opcjonalne: można do układu dołożyć przełącznik na jednym z przewodów, tak aby głośniczek wyłączał się po jego wciśnięciu. Następnie, jeżeli konstrukcja przełącznika na to pozwala, można zrobić tak, by głośniczek włączał się po wciśnięciu przełącznika.

Podłączamy buzzer pod pin sygnałowy i dołączamy przyciski – 15 minut

Podłączamy układ w sposób poniższy:



Buzzer, czyli brzęczyk, jest zapięty do pinu numer 3, oznaczonego PWM. Jest to pin **analogowy**, pozwalający na wysyłanie określonych przez nas wartości napięcia do brzęczyka. To powoduje, że możemy na brzęczyku wygrywać różne tony, zmieniając wielkość biegnącego do niego napięcia. Druga nóżka, w naszym wypadku czarna, jest zapięta do wspólnej linii GND, czyli uziemienia.

Używamy długich rzędów pinów z boku płytki stykowej, biegnących koło niebieskiej i czerwonej linii, do poprowadzenia linii zasilania PWR i uziemienia GND, po czym podłączamy do nich komponenty w zaznaczony sposób.

Piny na środku płytki stykowej, rozdzielone rowkiem, biegną prostopadle do pinów bocznych (czyli tych przy liniach niebieskiej i czerwonej). Zwracamy uwagę na ich zielone zakolorowanie w okolicach przycisków. Te piny, w liniach zgodnie z zakolorowaniami, są ze sobą połączone.

Przełączniki są jedną nóżką zapięte do PWR (pośrednio, przez listwę boczną), a drugą odpowiednio do pinów: 8, 9, 10. Naciśnięcie któregokolwiek przełącznika spowoduje zamknięcie obwodu, a więc dopływ prądu do odpowiadającego mu pina na płytce Arduino. Wzrost napięcia na tych pinach spowoduje że będziemy mogli tę wartość odczytać w programie.

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

Przypomnienie materiału, odtworzenie układu – 10 minut

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

Programowanie układu – 35 minut

Po podłączeniu płytki, przystępujemy do napisania programu który będzie ją obsługiwał. Chcemy, by każdy z przycisków – gdy zostanie naciśnięty – wydawał inny dźwięk. W tym celu zastosujemy trzy **instrukcje warunkowe „if”**, po jednej dla przycisku. Każda z nich będzie powodowała wysłanie prądu o różnym napięciu na pin analogowy numer 3, na którym jest głośnik. Użyjemy do tego funkcji **analogWrite**.

Będziemy wykorzystywać **zmienne**, czyli symboliczne słowa, do których przypisujemy w programie zmieniającą się wartość. Np. w naszym przypadku będą się zmieniały stany przycisków – raz będą włączone, raz wyłączone.

Żeby zdefiniować zmienną, musimy napisać jaki jest jej typ. Na początek użyjemy zmiennej która jest liczbą całkowitą, po angielsku **integer**, w skrócie – **„int”**:

int stanPrzełącznika1 = 0;

Ten przykład definiuje (czyli określa) zmienną i nadaje jej wartość początkową 0 za pomocą znaku „=”. Pojedynczy znak „=” zmienia wartości. Jeżeli chcemy sprawdzić, czy jakaś wartość jest równa innej, używamy PODWÓJNEGO znaku równości, czyli **„==”**.

Są też inne typy zmiennych, np. „long”, „float”, ale nimi nie będziemy się teraz zajmować. Chętni mogą je wpisać w wyszukiwarkę, np. anglojęzyczną Wikipedię, po

czym kliknąć w polską wersję językową w pasku po lewej stronie. Jest to bardzo dobra praktyka wyszukiwania informacji na temat programowania dla osób nie znających dobrze języka angielskiego.

Instrukcje warunkowe polegają na tym, że jeśli jakiś określony warunek zostanie spełniony, program wykona operację. Np. jeżeli nasza zmienna **stanPrzełącznika** jest równa **HIGH**, czyli na pin płynie prąd, będziemy chcieli wykonać jakąś operację. Po angielsku **„jeżeli"** to **„if"**.

Zwracamy uwagę na podwójny znak równości – jest on porównaniem dwóch wartości. Pojedynczy używamy do ich zmieniania! (wtedy wartość po lewej zostanie zmieniona na wartość po prawej stronie znaku).

```
if (stanPrzelacznika1 == HIGH) {
  // tu opisujemy co ma zostać wykonane
}
```

Funkcja **analogWrite** pozwala wysłać różne wartości napięcia poprzez wyznaczone do tego piny. Piny, które są obsługiwane przez tę funkcję, to te oznaczone PWM. Różnica z **digitalWrite** polega na tym, że digitalWrite może być tylko włączone, lub nie, wysyłając maksymalny dostępny prąd. analogWrite ma swoje minimalne i maksymalne wartości, które skaluje do wysyłanego prądu. Muszą się one mieścić w przedziale **od 0 do 255**.

Dla zainteresowanych, opis funkcji analogWrite jest dostępny w języku angielskim pod poniższym adresem:
https://www.arduino.cc/en/Reference/AnalogWrite

Polecamy również przestudiować w wolnym czasie dołączone do oprogramowania Arduino przykłady.

Nasz kod powinien wyglądać w sposób następujący:

```
void setup(){
  // najpierw definiujemy piny do których są podłączone przełączniki.
  // Ponieważ wysyłamy do nich informację przełącznikiem (HIGH albo LOW)
  // będą one wykorzystywane jako wejścia, a więc INPUT:
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);

  // następnie definiujemy pin na którym znajduje się głośnik
  pinMode(3, OUTPUT);
}

void loop(){
  // najpierw zdefiniujemy zmienne, jednocześnie przypisując do nich stan przycisków na początku pętli
  int  stanPrzelacznika1 = digitalRead(8); // czytamy stan z pinu 8
  int  stanPrzelacznika2 = digitalRead(9); // czytamy stan z pinu 9
  int  stanPrzelacznika3 = digitalRead(10); // czytamy stan z pinu 10

  // następnie dodamy dla każdego przycisku komendę warunkową.
  // przycisk 1:
  if ( stanPrzelacznika1 == HIGH){ // kiedy przełącznik 1 jest włączony
    // na pin analogowy 3 wysyłamy (255 dzielone na 3) = 85, czyli jedna trzecia pełnej mocy
    analogWrite(3, 85);
  }

  // przycisk 2:
  if ( stanPrzelacznika2 == HIGH){ // kiedy przełącznik 2 jest włączony
    // na pin analogowy 3 wysyłamy (255 dzielone na 3)*2 = 170, czyli dwie trzecie pełnej mocy
    analogWrite(3, 170);
  }

  // przycisk 3:
  if ( stanPrzelacznika3 == HIGH){ // kiedy przełącznik 3 jest włączony
    // na pin analogowy 3 wysyłamy pełną moc
    analogWrite(3, 255);
  }
}
```

Ładujemy kod na płytkę Arduino. Przy włączaniu pojedynczych przełączników każdy z nich powinien wydawać inny dźwięk.

Zadania dla uczniów:

Dodajcie więcej przełączników i zaprogramujcie je. Sprawcie, by buzzer grał dwa tony po wciśnięciu przycisku, jeden po drugim (np. zastosujcie komendę delay(100)).

Jeżeli moduły początkowe zostały wykonane przez grupę w założonym czasie, w tym miejscu następuje koniec zajęć. Jeżeli grupa dała radę wykonać je szybciej, możemy przejść do dalszego etapu.

Dalsze przykładowe zadania

if... else... – jeżeli... w przeciwnym razie...

W obecnym stanie, nasz program po wciśnięciu kilku przycisków na raz nie będzie się zachowywał w sposób pożądany. Jeżeli wciśniemy dwa lub więcej przycisków na raz, program będzie się starał zapisać dwie wartości na pin analogowy na raz.

To się oczywiście nie stanie, ponieważ program jest wykonywany w zapisanej przez nas kolejności, niemniej jednak nie jest to rozwiązanie ani pożądane, ani eleganckie. By tego uniknąć, możemy zmodyfikować nasz warunek tak, by zawierał klauzulę „else", czyli „w przeciwnym wypadku". Użycie jej spowoduje, że tylko pierwszy poprawny warunek zostanie wykonany, a reszta zignorowana. Zmodyfikowany kod będzie wyglądał w ten sposób:

```
void setup(){
  // najpierw definiujemy piny do których są podłączone przełączniki.
  // Ponieważ wysyłamy do nich informację przełącznikiem (HIGH albo LOW)
  // będą one wykorzystywane jako wejścia, a więc INPUT:
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);

  // następnie definiujemy pin na którym znajduje się głośnik
  pinMode(3, OUTPUT);
```

```
}

void loop(){
  // najpierw zdefiniujemy zmienne, jednocześnie przypisując do nich stan przycisków na początku pętli
  int  stanPrzelacznika1 = digitalRead(8); // czytamy stan z pinu 8
  int  stanPrzelacznika2 = digitalRead(9); // czytamy stan z pinu 9
  int  stanPrzelacznika3 = digitalRead(10); // czytamy stan z pinu 10

  // następnie dodamy dla każdego przycisku komendę warunkową.
  // przycisk 1:
  if ( stanPrzelacznika1 == HIGH){ // kiedy przełącznik 1 jest włączony
    // na pin analogowy 3 wysyłamy (255 dzielone na 3) = 85, czyli jedna trzecia pełnej mocy
    analogWrite(3, 85);
  }

  // przycisk 2:
  // dodajemy „else” - wykonaj tylko wtedy, jeżeli poprzedni warunek nie został spełniony
  else  if ( stanPrzelacznika2 == HIGH){ // kiedy przełącznik 2 jest włączony
    // na pin analogowy 3 wysyłamy (255 dzielone na 3)*2 = 170, czyli dwie trzecie pełnej mocy
    analogWrite(3, 170);
  }

  // przycisk 3:
  // dodajemy "else" - wykonaj tylko wtedy, jeżeli poprzedni warunek nie został spełniony
  else if ( stanPrzelacznika3 == HIGH){ // kiedy przełącznik 3 jest włączony
    // na pin analogowy 3 wysyłamy pełną moc
    analogWrite(3, 255);
  }
}
```

Przy pierwszym „if” nie dopisujemy „else” - jest to początkowy warunek, więc „else” dla niego po prostu nie istnieje.

Teraz przyciski są w hierarchii: jeżeli wciśniemy wszystkie trzy, zagra tylko pierwszy. Jeżeli wciśniemy drugi i trzeci, zagra tylko drugi.

Dodajemy warunki dla wielu przycisków wciśniętych na raz

Teraz możemy sami zadecydować, jakie wartości będą grane na buzzerze, jeżeli dwa przyciski zostaną wciśnięte na raz. Po prostu ustalamy osobne wartości dla wszystkich kombinacji, a jest ich niewiele, ponieważ mogą być wciśnięte tylko następujące kombinacje:

- przełącznik 1 oraz 2,
- przełącznik 2 oraz 3,
- przełącznik 1 oraz 3,
- przełącznik 1 oraz 2 oraz 3.

To „oraz”, którego użyliśmy powyżej w języku programowania Arduino, jest przedstawiane za pomocą podwójnego znaku &, czyli **„&&”**. Wygląda to tak:

```
if ( stanPrzelacznik1 == HIGH &&  stanPrzelacznika2 == HIGH){
  // wykonaj program
}
```

Natomiast ważna tu jest kolejność wykonywania czynności w programie: nie możemy tego dopisać gdziekolwiek.

Jeżeli napiszemy to na końcu programu, to wtedy program nigdy nie dojdzie do tego elementu, ponieważ wciśnięcie pojedynczego przycisku spowoduje odrzucenie naszego kodu przez „else”. A więc musimy spisać kod w odpowiedniej kolejności, zaczynając od 3 wciśniętych przycisków, przechodząc przez 2, aż do pojedynczych wciśnięć.

Ustalmy też wartości, jakie będziemy wysyłać do buzzera. Mamy trzy przyciski, trzy kombinacje „dwuprzyciskowe” i jedną kombinację trzyprzyciskową, co daje nam siedem rodzajów dźwięków. Do pinu analogowego możemy wysyłać maksymalną wartość 255. Tak więc 255/7=36,4285(...). Zaokrąglamy do liczby całkowitej. A więc wielkość, o którą będziemy zwiększali prąd na pinie analogowym to 36. Ustalmy teraz kolejność i wartości dla każdej kombinacji:

przełącznik1 = 36*1 = 36;
przełącznik1 +przełącznik2 = 36*2 = 72;
przełącznik1 +przełącznik3 = 36*3 = 108;
przełącznik2 = 36*4 = 144;
przełącznik2 +przełącznik3 = 36*5 = 180;
przełącznik3 = 36*6 = 216;
przełącznik1 + przełącznik2 + przełącznik3 = 36*7 = 252;

Pozostaje napisać to w formie kodu:

```
void setup(){
  // najpierw definiujemy piny do których są podłączone przełączniki.
  // Ponieważ wysyłamy do nich informację przełącznikiem (HIGH albo LOW)
  // będą one wykorzystywane jako wejścia, a więc INPUT:
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);

  // następnie definiujemy pin na którym znajduje się głośnik
  pinMode(3, OUTPUT);
}

void loop(){
  // najpierw zdefiniujemy zmienne, jednocześnie przypisując do nich stan przycisków na początku pętli
  int stanPrzelacznika1 = digitalRead(8); // czytamy stan z pinu 8
  int stanPrzelacznika2 = digitalRead(9); // czytamy stan z pinu 9
  int stanPrzelacznika3 = digitalRead(10); // czytamy stan z pinu 10

  // następnie dodamy dla każdego przycisku komendę warunkową.
  // na początek dla trzech przycisków:
  // kiedy przełącznik 1 oraz 2 oraz 3 są włączone
  if ( stanPrzelacznika1 == HIGH && stanPrzelacznika2 == HIGH && stanPrzelacznika3 == HIGH){
    // wartość z naszych wyliczeń to 252
    analogWrite(3, 252);
  }
```

```
  // następnie dla dwóch - 1 i 2:
  // kiedy przełącznik 1 oraz 2 są włączone
  else if ( stanPrzelacznika1 == HIGH && stanPrzelacznika2 == HIGH ){
    // wartość z naszych wyliczeń to 72
    analogWrite(3, 72);
  }

  // następnie dla dwóch - 1 i 3:
  // kiedy przełącznik 1 oraz 3
  else if ( stanPrzelacznika1 == HIGH && stanPrzelacznika3 == HIGH ){
    // wartość z naszych wyliczeń to 108
    analogWrite(3, 108);
  }

  // następnie dla dwóch - 2 i 3:
  // kiedy przełącznik 2 oraz 3
  else if ( stanPrzelacznika2 == HIGH && stanPrzelacznika2 == HIGH ){
    // wartość z naszych wyliczeń to 180
    analogWrite(3, 180);
  }

  // teraz zmieniamy wartości dla przycisków pojedynczych
  // przycisk 1,
  // dodajemy „else”
  else if ( stanPrzelacznika1 == HIGH){ // kiedy przełącznik 1 jest włączony
    // 36
    analogWrite(3, 36);
  }

  // przycisk 2:
  // dodajemy "else" - wykonaj tylko wtedy, jeżeli poprzedni warunek nie został spełniony
  else  if ( stanPrzelacznika2 == HIGH){ // kiedy przełącznik 2 jest włączony
    // 144
    analogWrite(3, 144);
  }

  // przycisk 3:
```

```
  // dodajemy "else" - wykonaj tylko wtedy, jeżeli poprzedni warunek nie został spełniony
  else if ( stanPrzelacznika3 == HIGH){ // kiedy przełącznik 3 jest włączony
    // 216
    analogWrite(3, 216);
  }
}
```

Przeanalizujmy teraz co się dzieje:

1. Sprawdzamy, czy wszystkie trzy przyciski są wciśnięte (1 moduł warunkowy). Jeżeli nie, to:
2. Sprawdzamy, czy dowolne dwa przyciski są wciśnięte (3 moduły warunkowe). Jeżeli nie, to:
3. Sprawdzamy, czy dowolny jeden przycisk jest wciśnięty (3 moduły warunkowe).

Pytania do uczniów:

- *Czy nasz kod będzie działał tak samo jeżeli pozycję 3 przeniesiemy na początek?*
- *Jeżeli nie, to dlaczego?* (jeżeli odpowiedź nie jest znana, należy przetestować modyfikując kod).

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.
Klasy I-III mogą wykonać zadanie przy użyciu programu mBlock opisanego w scenariuszu 18. Klasy IV-VI mogą wykonać zadania dodatkowe, jeżeli pozwoli na to czas i doświadczenie uczestników.

ZADANIE SPRAWDZAJĄCE UMIEJĘTNOŚCI ZDOBYTE PODCZAS ZAJĘĆ:

Uczeń / uczennica, pracując samodzielnie albo w dwu- lub trzyosobowym zespole buduje układ z wykorzystaniem Arduino, brzęczyka i przycisków. Zadanie polega na podpięciu przez uczniów układu, wytłumaczeniu, gdzie są piny sygnałowe (buzzera i przełączników), gdzie są piny zasilające, a gdzie uziemienie. Uczniowie powinni być w stanie wytłumaczyć własnymi słowami, co robi program, który wgrywaliśmy na płytkę, oraz wyjaśnić czym jest „if... else” („jeżeli... w innym przypadku”).

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

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

Wywiad z Imogen Heap, artystką budującą elektronikę do tworzenia muzyki
https://www.youtube.com/watch?v=ci-yB6EgVW4

Granie na puszkach napojów podłączonych do Arduino - dźwiękami i samplami:
https://www.youtube.com/watch?v=Ttm62RBdOuo

Czym jest brzęczyk, i jaka jest jego zasada działania:
https://pl.wikipedia.org/wiki/Brz%C4%99czyk

Zasady bezpieczeństwa w postępowaniu z modułami Arduino:
https://www.rugged-circuits.com/10-ways-to-destroy-an-arduino/

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).