AGH – UNIVERSITY OF SCIENCE AND TECHNOLOGY

Ardunino

Systemy wbudowane
grupa: wtorek 11:15

Marek Świątek, Gabriela Kowalik

Dokumentacja użytkownika

W tej części omówione zostaną: co to jest Ardunino, poszczególne elementy Ardunino, jakie tryby pracy oferuje urządzenie i jak uruchomić oraz posługiwać się nim w każdym z trybów.

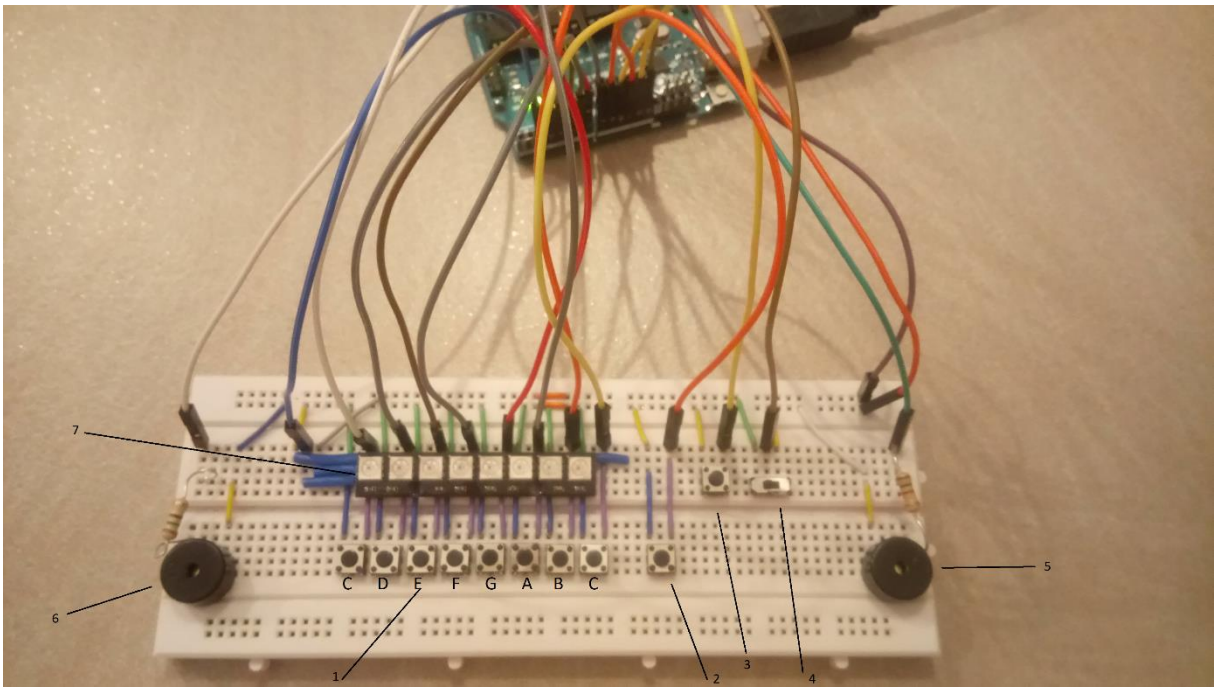
Co to jest Ardunino?

Płytka grająca Ardunino to idealne urządzenie do stawiania pierwszych kroków w świecie muzyki, wzorowane na instrumencie klawiszowym typu pianino, obsługujące jednoczesne granie kilku dźwięków.

Dzięki niej użytkownik w prosty sposób nauczy się grać dedykowane piosenki zapisane w pamięci urządzenia oraz będzie mógł sam tworzyć swoje własne kompozycje czy ćwiczyć poznane utwory.

Z czego składa się Ardunino?

Poniżej przedstawiono i opisano elementy Ardunino:



Zdjęcie 1 – model Ardunina

1. **Panel klawiszy**, osiem przycisków, każdy reprezentujący inny dźwięk z oktawy CDEFGABC (kolejność jak na zdjęciu)
2. **Przycisk podniesienia oktawy**, działający z jednocześnie wciśnięty z przyciskiem z panelu klawiszy, podnosi oktawę wszystkich granych dźwięków
3. **Przycisk zmiany piosenki**, aktywny jedynie w trybie nauki, w trybie zabawy jest nieaktywny
4. **Przełącznik trybów zabawy/nauki**, służy do przełączania między dwoma trybami oferowanymi przez urządzenie

5. **Prawy głośnik**
6. **Lewy głośnik**
7. **Listwa diod LED**, 8 diod LED, zapalających się w zależności od trybu pracy; każda z diod jest przypisana odpowiedniemu tonowi i świeci się wg niniejszego schematu

Dioda (od lewej)	Dźwięk	Kolor tonu niskiego (oktawa piąta)	Kolor tonu wysokiego (oktawa szósta)
1	c	czerwony	biały
2	d	różowy	
3	e	pomarańczowy	
4	f	żółty	
5	g	zielony	
6	a	cyjanowy	
7	b	niebieski	
8	C	fioletowy	

Jakie tryby oferuje Ardunino?

Ardunino oferuje dwa tryby pracy, opisane poniżej:

1. **Tryb nauki**

   W tym trybie użytkownik ma możliwość nauczenia się sekwencji klawiszy potrzebnych do odtworzenia kilku melodii zapisanych w pamięci urządzenia.

   Nauka polega na obserwacji sekwencji zapalania się diod na listwie (nr 7, *zdjęcie 1*) oraz wciskaniu odpowiadających im przycisków.

   Dźwięk zostanie zagrany wtedy, gdy użytkownik prawidłowo wciśnie przycisk w czasie świecenia się odpowiadającej mu diody oraz uwzględni wciśnięcie (bądź nie, w zależności od koloru zapalonej diody) pedała zmiany oktawy. Jeżeli użytkownik wciśnie przycisk, którego odpowiadająca dioda nie jest zapalona lub spróbuje zagrać dźwięk o nieprawidłowej wysokości, dźwięk nie zostanie zagrany.

   Gdy sekwencja świateł dla piosenki zakończy się, urządzenie zaczyna ją powtarzać. Użytkownik ma możliwość zmiany sekwencji na kolejną zapisaną w pamięci urządzenia.

2. **Tryb zabawy**

   W tym trybie użytkownik ma możliwość tworzenia własnych kompozycji dźwiękowych czy ćwiczenia poznanych kompozycji bez blokady dźwięków, tak jak to było w trybie nauki, oraz ograniczeń czasowych.

   Użytkownik sam decyduje jakie dźwięki chce zagrać oraz steruje ich wysokością i długością.

   Sekwencja zapalanych świateł jest ustalona przez kolejność i czas trwania wciskania przycisków przez użytkownika.

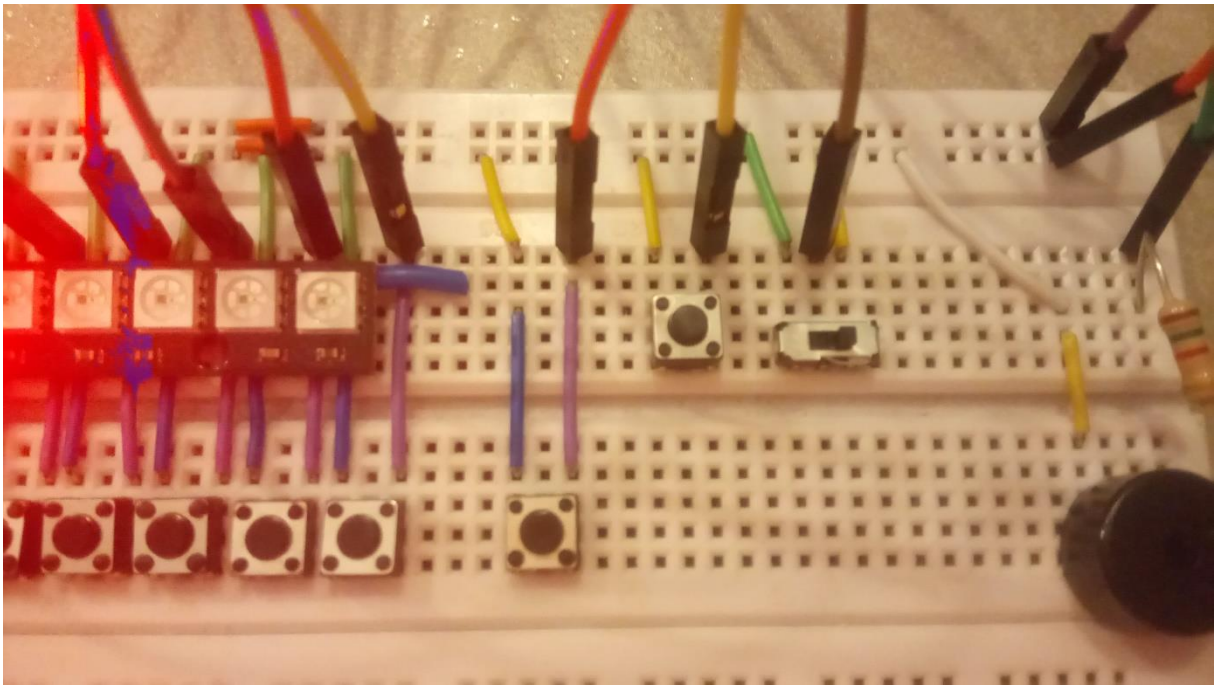
Jak uruchomić Ardunino?

Aby uruchomić urządzenie, należy podłączyć urządzenie do zasilania przez port USB (5 V), na przykład do komputera czy Power Banku, bądź wejście DC (7 – 12 V).

Jak posługiwać się trybami Ardunino?

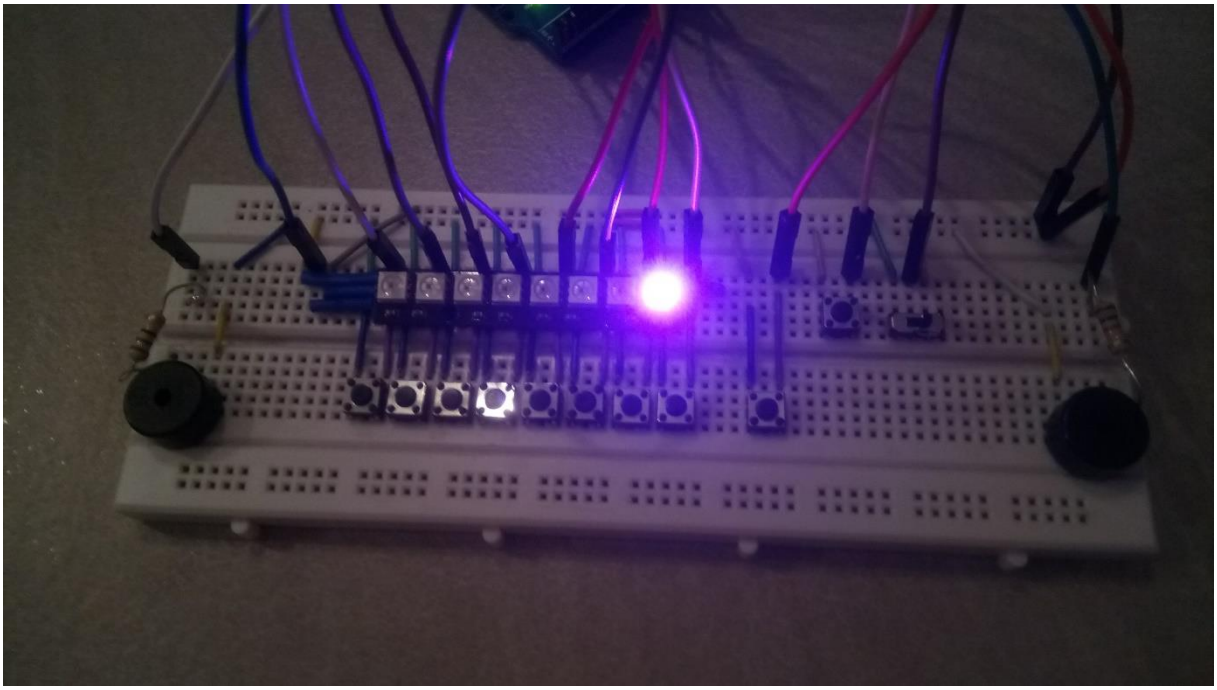
Przejście do trybu nauki:

Aby przejść do trybu nauki, należy przełączyć przełącznik zmiany trybów (nr 4, *zdjęcie 1*) na stan pokazany na poniższym zdjęciu.

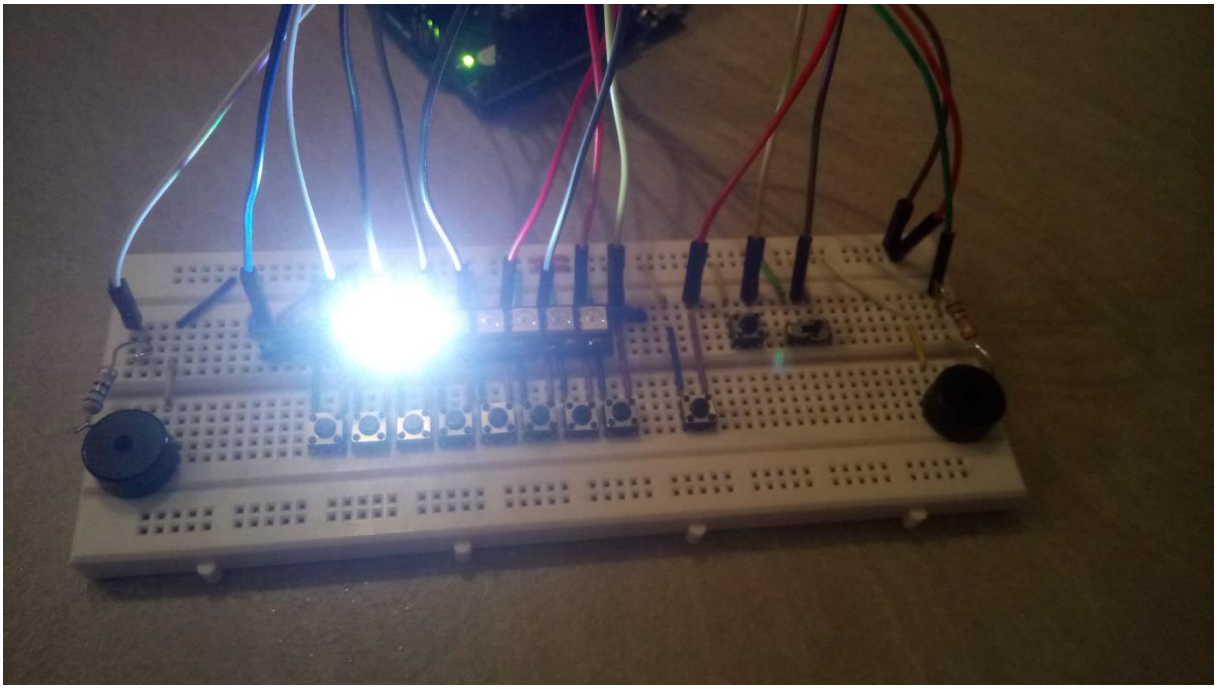


Zdjęcie 2: przełącznik w trybie nauki

Urządzenie zacznie zapalać sekwencję diod wybranej piosenki. Aby zagrać dźwięk należy wcisnąć przycisk odpowiadający zapalonej diodzie oraz przycisk podniesienia oktawy (nr 2, *zdjęcie 1*), w zależności od tego czy jest to dioda niskiej (*zdjęcie 3*) czy wysokiej (*zdjęcie 4*) oktawy. Aby zmienić piosenkę, należy wcisnąć przycisk zmiany piosenki (nr 3, *zdjęcie 1*).



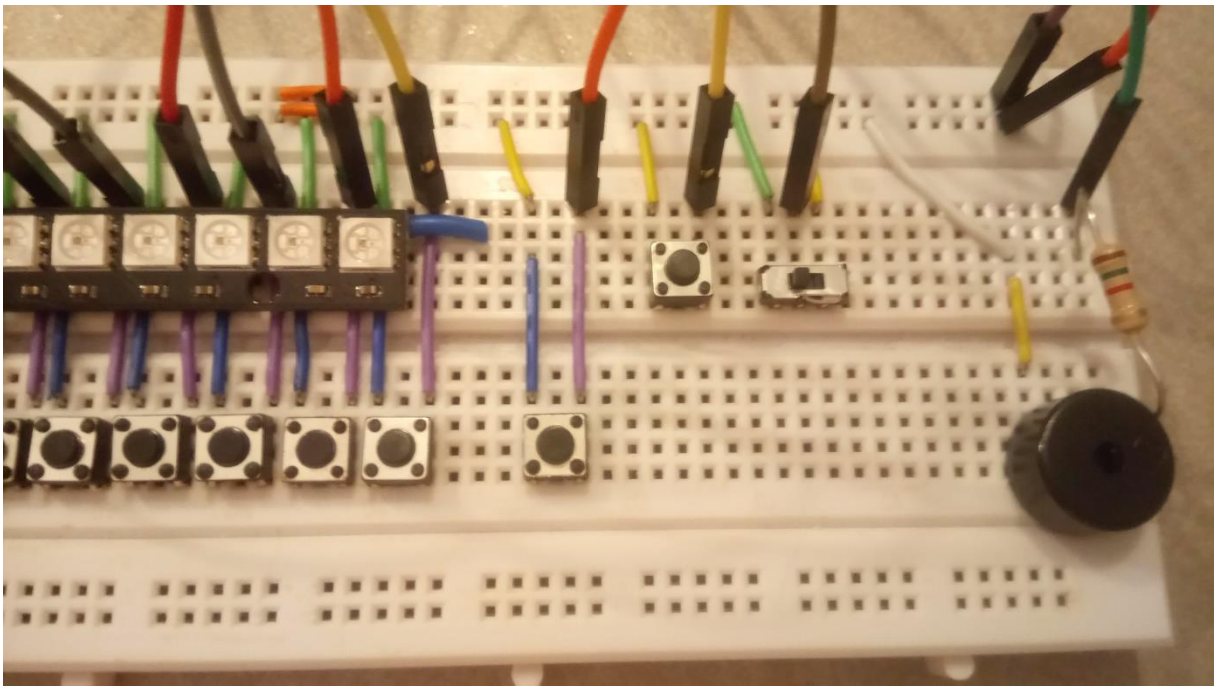
Zdjęcie 3: zapalona dioda niskiej oktawy w trybie nauki



Zdjęcie 4: zapalona dioda wysokiej oktawy w trybie nauki

Przejście do trybu zabawy:

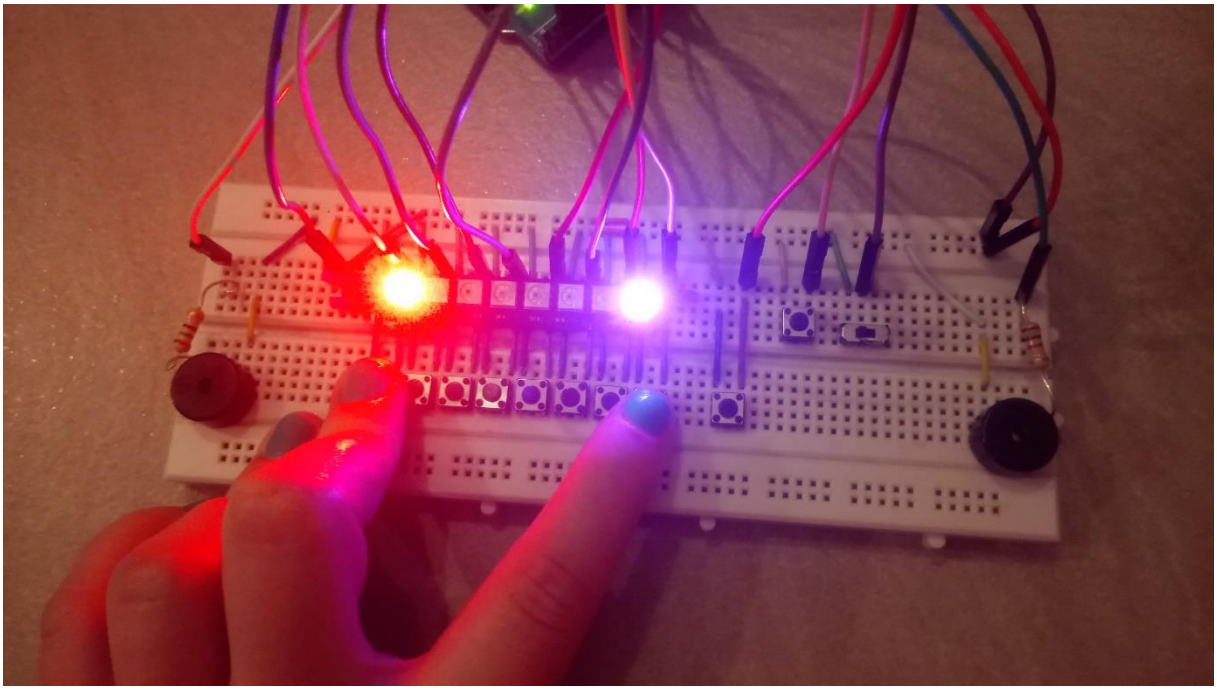
Aby przejść do trybu zabawy, należy przełączyć przełącznik zmiany trybów (nr 4, *zdjęcie 1*) na stan pokazany na poniższym zdjęciu.



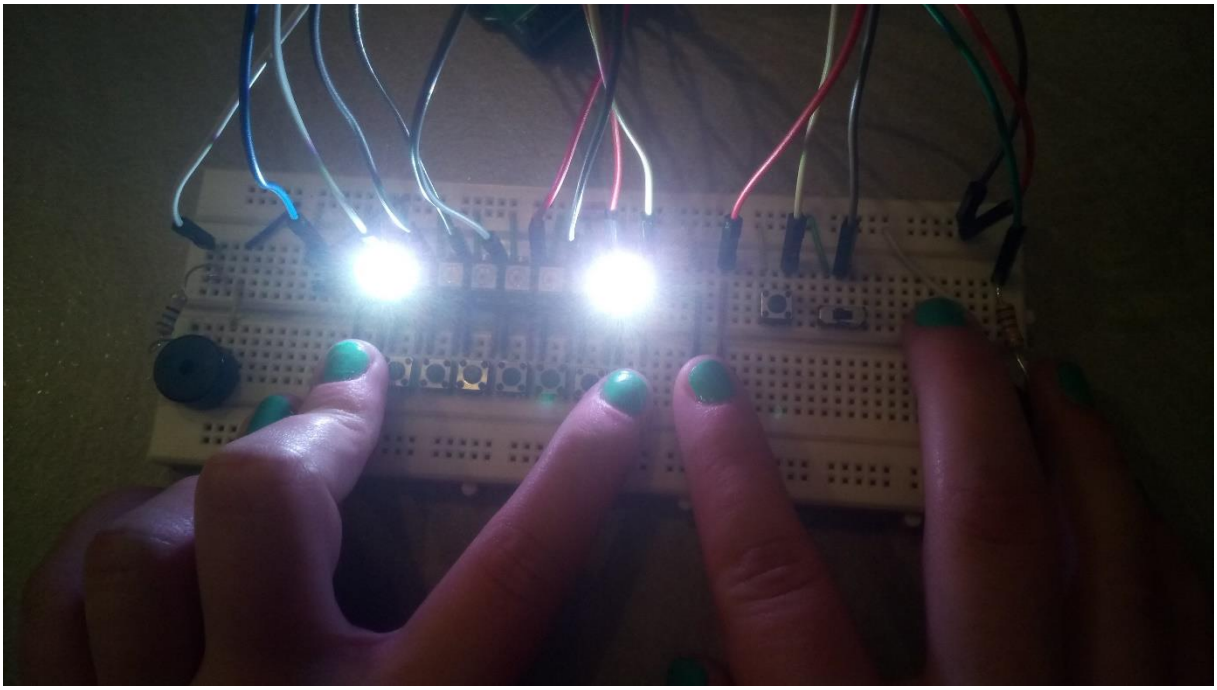
Zdjęcie 5: przełącznik w trybie zabawy

Aby zagrać wybrany dźwięk, należy wcisnąć przycisk odpowiadający danemu dźwiękowi oraz przycisk podniesienia oktawy (nr 2, *zdjęcie 1*), w zależności od pożądanej przez użytkownika wysokości dźwięku. Po wciśnięciu przycisku, dioda odpowiadająca dźwiękowi zostanie zapalona na kolor zależny od wysokości granego dźwięku.

W tym trybie przycisk zmiany piosenki (nr 3, *zdjęcie 1*) jest nieaktywny – wciśnięcie go nie spowoduje żadnych akcji.



Zdjęcie 6: tryb zabawy bez wciśniętego przycisku podniesienia oktawy



Zdjęcie 7: tryb zabawy z wciśniętym przyciskiem podniesienia oktawy

Dokumentacja techniczna

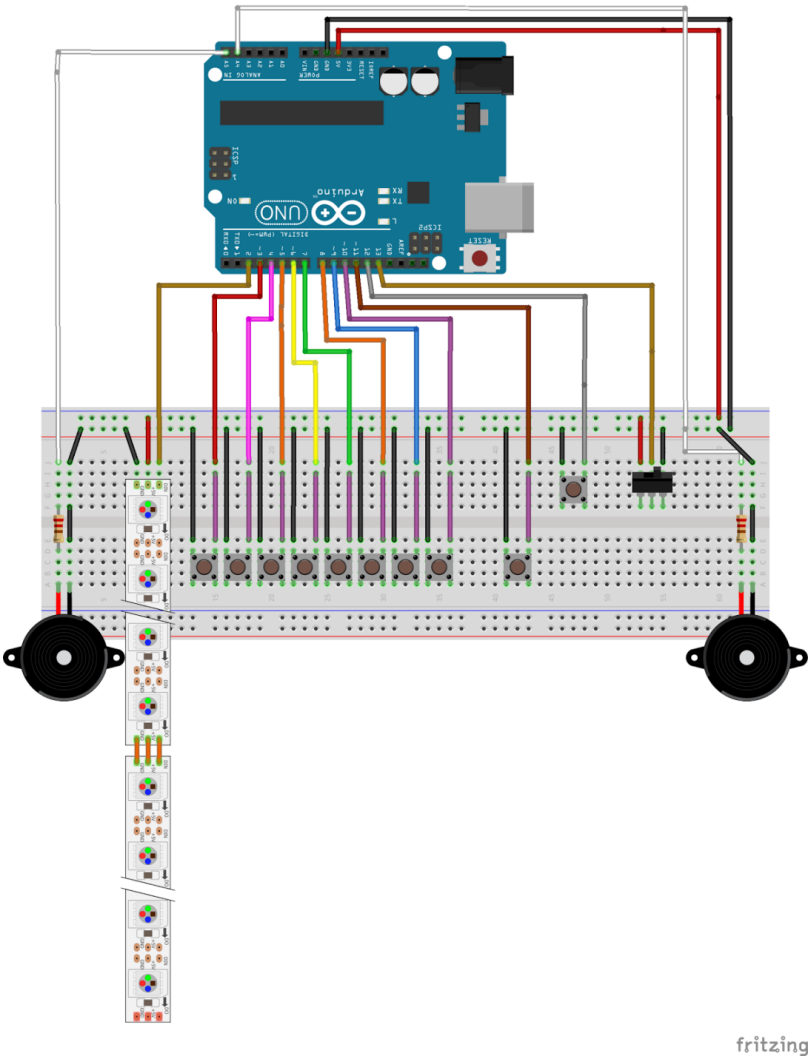
W tej części omówione zostaną: opis złożonego układu, instalacja programu na płytce oraz opis struktury programu, struktur, klas i ich metod, funkcji oraz stałych wykorzystywanych w programie.

Schemat układu.

Do złożenia układu wykorzystano następujące komponenty:

- płytka Arduino UNO rev3 ard-01060
- płytka stykowa
- przewody połączeniowe
- 2 x brzęczyk piezoelektryczny
- 10 x przycisk
- suwak 2-stanowy
- taśma LED (8 diod Neopixel ws2812b)
- 2 rezystory 220 Ohm

Układ jest złożony na podstawie poniższego schematu:



Schemat 1: Schemat układu

Brzęczyki są podpięte do wejść A4 i A5 (pomiędzy brzęczykami, a wyjściami są wpięte rezystory), drugie wyjścia brzęczyków są podłączone do uziemienia.

Przyciski są podpięte do wejść od 3 do 12. Nie są potrzebne rezystory, ponieważ przy ustawianiu wejść skorzystano z trybu rezystora podciągającego.
Osiem przycisków (pierwsze osiem od lewej na *Schemat Układu*) jest klawiaturą, kolejny odpowiada za pedał zmiany oktawy i kolejny za zmianę piosenek w Trybie Nauki. Każdy przycisk jest podpięty do uziemienia.

Do wejścia 13 jest podpięty suwak dwustanowy. Analogicznie jak do przycisków skorzystano z rezystora podciągającego. Suwak jest podpięty również do uziemienia i do 5V.

Do wejścia 2 jest podpięta taśmy 8-ledowa (na schemacie znajdują się 2 podłączone szeregowo taśmy 4-ledowe). Taśma jest podpięta również do uziemienia i do 5V.

Instalacja programu na płytce.

Program powinien być wgrany według standardowego schematu wgrywania prostych programów na płytkę Arduino.
Do projektu należy załączyć dwie zewnętrzne biblioteki:

- Adafruit Neopixel, do pobrania z: github.com/adafruit/Adafruit_NeoPixel
- Tone, do pobrania z: github.com/bhagman/Tone

Struktura programu.

Wykorzystano podejście obiektowe - w programie kluczowe i skomplikowane elementy są reprezentowane przez obiekty.

W projekcie istnieją pomocnicze klasy **MyColor** oraz struktura **Colors**.
Fizyczne elementy reprezentują następujące klasy:

- **Strip** - obsługująca taśmę led (włączanie i włączanie diod),
- **ToneKeyboard** - obsługująca klawiaturę,
- **Speaker** - obsługująca głośniki

Przyciski: zmiany oktawy, zmiany piosenki oraz przełącznik zmiany trybu ze względu na proste użycie (jedynie odczyt stanu) nie zostały zmapowane na obiekty.

Za tryb nauki odpowiada klasa **Tutor**.

Opis poszczególnych elementów programu.

Klasa MyColor

Odpowiada za wyświetlane kolory.

Metody:

SimpleColor::SimpleColor(const int red, const int green, const int blue)
Konstruktor klasy; przyjmuje jako parametry kolejne wartości zapisu koloru w RGB.

const int SimpleColor::getRed()
Zwraca natężenie czerwonego w kolorze.

const int SimpleColor::getGreen()
Zwraca natężenie zielonego w kolorze.

const int SimpleColor::getBlue()
Zwraca natężenie niebieskiego w kolorze.

Struktura Colors

Zawiera tablicę kolorów odpowiadających poszczególnym tonom niskim, kolor dla tonu wysokiego oraz kolor dla braku tonu.

Klasa Strip

Klasa Strip odpowiada za obsługę taśmy LED. Korzysta ze struktury Colors.

Metody:

void Strip::setStrip()
Odpowiada za zainicjowanie obsługi taśmy

void Strip::turnOnLed(int led, bool high)
Włącza diodę o nr *led.* Jeśli high jest prawdziwe, ustawia kolor diody na TURNED_HIGH, w przeciwnym przypadku odpowiednią wartość z tablicy TURNED_LOW.

void Strip::turnOffLed(int led)
Wyłącza diodę o numerze *led*.

Klasa Strip korzysta z zewnętrznej biblioteki Adafruit Neopixel.

Klasa Speaker

Klasa Speaker odpowiada za obsługę brzęczyków. Są one obsługiwane wspólnie - dźwięk (o takiej samej wysokości) jest jednocześnie nadawany na oba brzęczyki.

Metody:

void Speaker::setSpeaker(const int pinLeft, const int pinRight)
Odpowiada za zainicjowanie obsługi głośników. W parametrach podawane są numery pinów do których podpięte są głośniki.

void Speaker::play(int tone)
Powoduje, że na obu głośnikach zaczyna grać dźwięk o wysokości *tone*. W metodzie zastosowano opóźnienie, które pozwala na iluzję odtwarzania paru dźwięków naraz (bez „szarpanych" zmian dźwięku).

void Speaker::stop()
Przerywa odtwarzanie obecnie nadawanego dźwięku.

Speaker korzysta z zewnętrznej biblioteki Tone.

Klasa ToneKeyboard

Odpowiada za rozpoznawanie, który przycisk został wciśnięty - czyli również za to jaki ton ma być zagrany lub który LED ma zostać zaświecony (wynika to z bezpośredniego powiązania przycisków z diodami i z powiązania przycisków z dźwiękami z wykorzystaniem tablic dźwięków niskich i wysokich.

Aby ułatwić translację pomiędzy pinami do których podpięte są przyciski, a przyciskami dodano strukturę **LedToPinMap** mapującą numer pinu na numer przycisku/ledu.

Metody:

void ToneKeyboard::setKeyboard(int buttons[])
Odpowiada za wypełnienie mapy powiązań pomiędzy pinami i przyciskami oraz za inicjalizację wejść do których podłączone są przyciski klawiatury.

const int ToneKeyboard::getTone(int button, bool isHigh)
Zwraca wysokość dźwięku powiązanego z danym przyciskiem.
Jeśli wartość is_high jest prawdziwa, zwraca dźwięk z wyższej oktawy - w przeciwnym przypadku dźwięk z oktawy niższej.

const int ToneKeyboard::getLed(int button)
Zwraca numer diody led powiązanej z przyciskiem, podpiętym pod pin nr *button.*

const int ToneKeyboard::getButton(int led)
Zwraca numer pin do którego jest podpięty przycisk powiązany z diodą o numerze *led.*

Klasa Tutor

Klasa Tutor odpowiada za obsługę Trybu Nauki. Zapisane są w niej predefiniowane piosenki. Do ich przechowywania używana jest struktura **Grip**:

```
struct Grip{
  bool high;
  char tab;
  long duration;
};
```

high - jeśli wartość jest prawdziwa, to znaczy że dźwięk ma być zagrany z wciśniętym pedałem poniesienia oktawy
tab - jest maską definiującą który klawisz ma być aktualnie wciśnięty, np. wartość 1010001b oznacza że przycisk nr 1, 3, 8 mają być wciśnięte
duration - czas trwania grania danego chwytu (konfiguracji tab-high)

Metody:

void Tutor::setTutor()
Odpowiada za ustawienie konfiguracji obiektu Tutor - inicjalizowana jest tablica piosenek.

void Tutor::nextSong()
Przełącza aktualną piosenki na kolejną w tablicy piosenek, w przypadku gdy obecna piosenka jest ostatnią, to następną będzie pierwszą z tablicy piosenek.

void Tutor::teach()
Rozpoczyna naukę gry piosenki. W przypadku zakończenia się piosenki, zaczyna uczyć obecnej piosenki od nowa. Aby zacząć naukę nowej piosenki, trzeba najpierw wywołać nextSong().

bool Tutor::getNote(bool* notes)
Nadpisuje tablicę *notes* maską aktualnie granego akordu, przykładowo: jeśli maska wynosi 1001000b, to do notes będzie przypisana tablica {true, false, false, true, false, false, false, false}.
Parametrem wejściowym jest wskaźnik do ośmioelementowej tablicy zmiennych typu bool.
getNote zwraca wartość false jeśli piosenka się skończyła - wartość true w przeciwnym przypadku.

bool Tutor::isHigh()
zwraca wartość true w przypadku, gdy aktualny dźwięk powinien być zagrany z wciśniętym pedałem - false w przeciwnym przypadku

Funkcja setup

W tej funkcji ustawiane są wszystkie elementy programu (za pomocą funkcji set_ zdefiniowanych w każdej z klas). Ponadto inicjalizowane są trzy wejścia:

```
pinMode(modeSwitchPin, INPUT_PULLUP
pinMode(switchSongPin, INPUT_PULLUP);
pinMode(octaveSetterPin, INPUT_PULLUP);
```

Są to piny odpowiedzialne kolejno za suwak zmiany trybu, przycisk zmiany piosenek i pedał zmiany oktawy.

Funkcja loop

W głównej pętli programu rozpoznawany jest bieżący tryb oraz wykorzystywany Tutor (w trybie nauki)

Pseudokod głównej pętli:

```
Loop():
  If Tryb zabawy then
    pianoMode w trybie zabawy
  else
    if przycisk zmiany piosenki jest włączony then
       zmień obecną piosenkę
    if nie zacząłeś uczyć piosenki then
       zacznij uczyć obecną piosenkę
    pianoMode w trybie Nauki z aktualnie uczonym akordem
```

Funkcja switchSong

Funkcja odczytuje stan przycisku zmiany piosenki, jeżeli został wciśnięty to zmienia piosenkę w Tutorze.

Funkcja pianoMode

Funkcja, która odpowiada za sprawdzenie po kolei przycisków klawiatury. Dla każdego przycisku wywołuje funkcję *checkButton.* W Trybie Nauki nie są sprawdzane przyciski, których nie ma w obecnie granym akordzie.

Funkcja checkButton

Funkcja odpowiadająca obsługę przycisku nr *i*. W Trybie Nauki zapala diodę led nad odpowiednim przyciskiem oraz odfiltrowuje źle zagrane dźwięki (z wciśniętym lub nie wciśniętym pedałem zmiany oktawy - zależnie od aktualnie uczonego akordu). W trybie

zabawy odpowiada za zaświecenie odpowiedniej diody LED i zagranie odpowiedniego dźwięku.

Pseudokod:

```
checkButton(i, trybNauki?):
  if trybNauki? then
     zapal led odpowiadający przyciskowi i

  if przycisk i wciśnięty then

     if trybNauki? and stan pedała == wysokość uczonego dźwięku then
        zagraj dźwięk odpowiadający przyciskowi i
     elif not trybNauki? then
        zagraj dźwięk odpowiadający przyciskowi i
        zapal diodę led odpowiadającą przyciskowi i

  else
     przestań grać dźwięk
     if not trybNauki? then
        wyłącz diodę led odpowiadającą przyciskowi i
```

Stałe wykorzystywane w programie.

Stałe odpowiadające za numery pinów do których podpięte są poszczególne komponenty Arduino:

```
const int buzzerRightPin = A4;
const int buzzerLeftPin = A5;
const int modeSwitchPin = 13;
const int switchSongPin = 12;
const int octaveSetterPin = 11;
const int notes[] = { 3, 4, 5, 6, 7, 8, 9, 10 };
const int diodePin = 2;
```

Ustawienie odpowiednich wejść odpowiada wejściom na schemacie układu.

Ponadto, w programie użyto następujących stałych:

```
const int TONES_AMOUNT = 8;
 const int SONGS_AMOUNT = 3;
```

TONES_AMOUNT określa liczbę dźwięków które obsługujemy (jest też wykorzystywana przy kolorach oraz mapowaniu przycisków na piny).
SONGS_AMOUNT jest stałą określającą liczbę piosenek na urządzeniu.

Dokumentacja procesu

W tej części omówione zostaną: opis założeń projektu, harmonogram prac oraz opis problemów napotkanych podczas budowy Ardunino oraz ich rozwiązania, a także podsumowanie i możliwości dalszego rozwoju projektu.

Założenie i krótka charakterystyka projektu.

Celem prac było wykonanie modelu minipianina z wykorzystaniem płytki Arduino. Dodatkowymi zaplanowanymi funkcjonalnościami były: wizualizacja granych dźwięków (wykonana za pomocą łańcucha LED) i Tryb Nauki pozwalający na opanowanie instrumentu w mistrzowskim stopniu (LEDy świecące nad odpowiednimi klawiszami w czasie grania zapisanej w pamięci melodii).

Bazowa wersja projektu zawiera 8 klawiszy reprezentujących oktawę CDEFGABC, pedał do zmiany oktawy (w postaci przycisku) oraz dwa dodatkowe przełączniki pozwalające na przejście w Tryb Nauki i zmianę piosenki w nim. Na płytce jest dołączona również listwa LED (8 diod RGB). Każdy ton jest reprezentowany przez inną diodę i kolor.

Obsługa audio jest zrealizowana za pomocy dwóch brzęczyków piezoelektrycznych (grających unisono).

Harmonogram Prac:

Data	**Wykonana praca:**
14.11.2017	Testowanie poszczególnych komponentów Ardunino
21.11.2017	Złożenie klawiatury i dodanie obsługi głośników
28.11.2017	Dodanie obsługi taśmy LED
05.12.2017	Refactor kodu przed dodaniem Trybu Nauki
12.12.2017	Wykonanie Trybu Nauki
19.12.2017	Przetłumaczenie i wgranie piosenek dla Trybu Nauki
09.01.2017	Poprawki, refactor oraz zmiana kodu pod kątem zużywanej pamięci

Napotkane problemy.

1. **Problem z brzęczykiem.**

   Do wykonania Ardunina, początkowo wybrano brzęczyk piezoelektryczny, który jest standardowo dołączany do zestawów edukacyjnych Arduino. Niestety brzęczyk nie reagował na wywołanie standardowej funkcji *tone*.

   Próby rozwiązania problemu:

   a) zmianę brzęczyka na inny model - było to działające rozwiązanie, lecz z powodu niestabilności podpięcia nowego głośnika i urażonej dumy projektantów, rozwiązanie to odrzucono
   b) wykorzystanie innego brzęczyka, ale tego samego modelu, w celu weryfikacji czy testowany głośnik nie był zepsuty. Okazało się jednak, że oba głośniki nie odtwarzały dźwięków. Dzięki tej próbie doszedł pomysł na wykorzystanie dwóch głośników na płytce, dzięki czemu została zwiększona głośność Ardunino
   c) znalezienie innego sposobu obsługi głośniczków. Skorzystano w tym celu z biblioteki znalezionej na Github: Tone (https://github.com/bhagman/Tone). Zdecydowano się na to rozwiązanie, ponieważ pozwoliło ono obsługiwać dwa razy więcej głośników, niż korzystanie ze standardowego tone. Dodatkowym atutem tego rozwiązania jest obsługa jednoczesnego grania kilku dźwięków.

2. **Problem grania akordów.**

   Jednym z problemów, z którym się zmierzono, była obsługa grania paru dźwięków jednocześnie.

   Wpłynął on następująco na projekt:

   a) nie można było zagregować kilku przycisków - dlatego zastosowano podejście jeden pin - jeden przycisk. Aby nie ograniczać możliwości instrumentu zdecydowano się na dodanie pedała, który pozwolił granie w zakresie dwóch oktaw (wciśnięty podwyższa wysokość dźwięku o oktawę). Pierwotnym założeniem było wykorzystanie w postaci pedała przełącznika trójstanowego, dzięki któremu można byłoby jeszcze bardziej rozbudować możliwości sprzętu - niestety przy zbyt emocjonalnym wykonywaniu utworów był zbyt mało stabilny i wypadał.
   b) podstawowa funkcja używana do obsługi głośniczków *tone*, nie pozwala na granie paru dźwięków na raz. Aby poszerzyć możliwości instrumentu skorzystano z biblioteki Tone z Githuba. Pozwala ona na zagranie paru tonów na raz. Niestety i tu napotkano ograniczenie - metody z tej biblioteki korzystają z timerów mikrokontrolera (w tym przypadku ATmega z timerami 2, 1, 0; umożliwiający zagranie dwóch - w porywach trzech, dźwięków jednocześnie). Aby obejść to kolejne ograniczenie kod został zmieniony tak, aby pozwolił danemu dźwiękowi się wybrzmieć: dodano opóźnienie 100ms między dźwiękami. W ten sposób

osiągnięto iluzję akordów oraz wygładzono trochę dźwięk – pierwotnie szybkie zmiany wysokości powodowały nieprzyjemny wibrujący efekt.

c) zdublowanie głośników – na początku wydawało się, że może to w jakiś pomóc w poradzeniu sobie z tym problemem - niestety była to złudna nadzieja. Pomyłka okazała się na tyle szczęśliwa, że zwiększyła końcową głośność dźwięków granych przez Ardunino.

3. **Problem z pamięcią:**

   Z powodu wyboru obiektowego podejścia do problemu i dużego rozmiaru piosenek (predefiniowanych specjalnie dla trybu nauki) pojawiły się problemy z rozmiarem programu. W pewnym momencie przekroczył on ilość pamięci dynamicznej urządzenia (SRAM - 2048B). Główny problem stanowiły piosenki, definiowane jako tablice struktur ***Grip***.

   Rozważano dwa podejścia do rozwiązania problemu:

   a) przeniesienie piosenek do pamięci programu - pomysł odrzucony ze względu na trudną, nieintuicyjną implementację oraz brak materiałów związanych z tym szczególnym problemem
   b) odchudzenie ***Grip***'a:
      Zauważono, że można odwzorować chwyt (tab - jakie przyciski mają być naciśnięte) na 8 bitach (char) zamiast na 8 bajtach. Zmieniono również wielkość zmiennych duration z long na short - pozwalających przechowywać czas do 32 768 ms - co i tak jest praktycznie za długim czasem na jeden akord. Uzyskano dzięki temu 4,25 krotne zmniejszenie ilości zajmowanej pamięci.

   Drugie rozwiązanie było w pełni zadawalające – pozostało bardzo dużo wolnej pamięci dynamicznej. Jednakże warto podkreślić, że przeniesienie dużych tablic do pamięci programu jest znacznie lepszym i pewniejszym rozwiązaniem.

   Aktualny stan pamięci:
   Pamięć programu: 9604B/32256B (29%)
   Pamięć dynamiczna: 656B/2048B (32%)

Możliwości dalszego rozwoju.

Możliwe dalsze zmiany, nowe funkcjonalności:

- odciążenie pamięci dynamicznej przez przeniesienie piosenek do pamięci programu,
- dodanie obsługi większego zakresu nut (zmiana sposobu realizacji pedała),
- odejście od realizacji Ardunino na przyciskach - tutaj ciekawym kierunkiem rozwoju jest użycie klawiszy dotykowych,
- stworzenie sposobu wgrywania piosenek, np. poprzez zapamiętywanie w pamięci dynamicznej aktualnie granej piosenki,
- dodanie poziomów trudności Trybu Nauki,
- rozbudowa klawiatury o kolejne przyciski,

- odtwarzanie melodii na zewnętrznym głośniku

Podsumowanie projektu

Niestety wbrew oczekiwaniu twórców płytka Arduino nie jest idealną bazą dla instrumentów mogących podbić światowy rynek muzyczny. Głównym problemem jest brak „prawdziwych" analogowych wyjść - wyjścia PWM nie dają wystarczająco gładkiego dźwięku.

Innym poważnym problemem jest utrudniona możliwość grania kilku dźwięków naraz (na szczęście rozwiązana w nowszych płytkach, które są oparte na mikrokontrolerze ATmega 1280, posiadającym aż 6 timerów).

Praca nad projektem - z początku wydającym się łatwym, stawiała przed twórcami dużo wymagających i ciekawych problemów. Większość z nich wynikała z całkowitego nieprzystosowania Arduino do celów muzycznych.

Nie zmienia to jednak faktu, że rozwiązywanie problemów oraz konstrukcja projektu pozwoliło na zapoznanie się z płytką Arduino oraz sposobami budowania układów elektronicznych i tworzenia prostych systemów wbudowanych i było świetną zabawą.