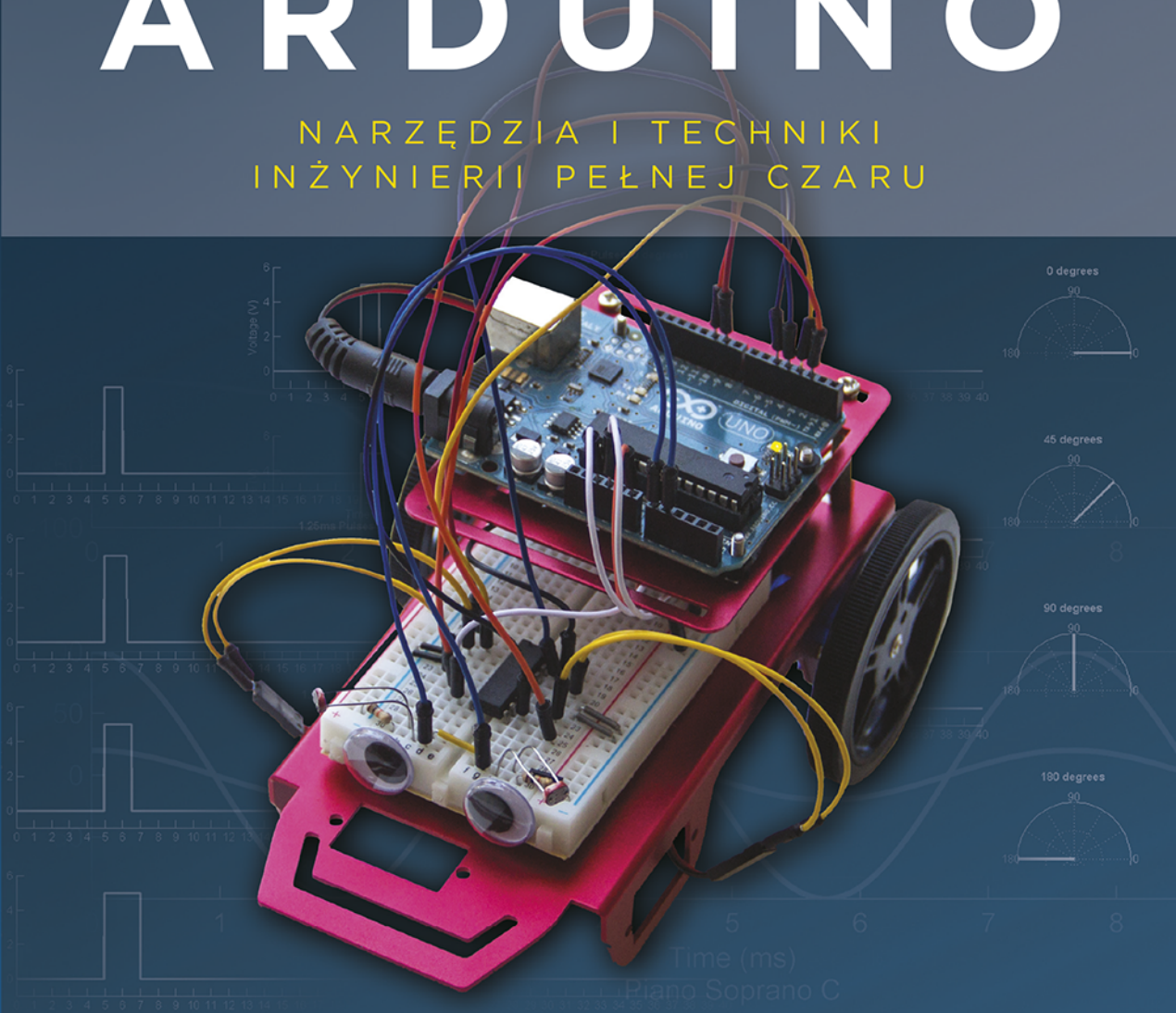
WYDANIE II

JEREMY BLUM

Nowe kolorowe zdjęcia i wykresy

ODKRYWANIE ARDUINO®

NARZĘDZIA I TECHNIKI INŻYNIERII PEŁNEJ CZARU



WILEY

Helion

Tytuł oryginału: Exploring Arduino: Tools and Techniques for Engineering Wizardry, 2nd Edition

Tłumaczenie: Anna Mizerska

ISBN: 978-83-283-6923-8

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/odkar2.zip

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/odkar2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

CZĘŚĆ IV. ODKRYWANIE BARDZIEJ ZAAWANSOWANYCH MOŻLIWOŚCI ORAZ ŁĄCZENIE FUNKCJI

CZĘŚĆ V. KOMUNIKACJA BEZPRZEWODOWA

Rozdział

6

Generowanie dźwięków i tworzenie muzyki

Co będzie potrzebne w tym rozdziale

- Arduino Uno lub Adafruit METRO 328,
- przewód USB (Uno — typ A-B, METRO — typ A-micro B),
- duża płytka stykowa (800 otworów) lub średnia płytka stykowa (400 otworów),
- różne przewody połączeniowe,
- 5 przycisków,
- rezystory 220 Ω,
- 5 rezystorów 10 kΩ,
- potencjometr 10 kΩ,
- głośnik 8 Ω.

Ludzie mają pięć zmysłów. Jak się pewnie domyślasz, nie będziesz używał zmysłu smaku, korzystając z elektroniki. Lizanie Arduino jest złym pomysłem. Podobnie jest ze zmysłem węchu. W zasadzie, jeżeli czujesz elektronikę, to znaczy, że prawdopodobnie coś się pali (i powinieneś przerwać to, co robisz). Pozostaje tylko zmysł dotyku, wzroku i słuchu. Stosowałeś już potencjometry i przyciski, które wykorzystują Twój zmysł dotyku, oraz podłączałeś diody LED, które oddziałują na Twój wzrok. Co powiesz na to, abyśmy teraz wykorzystali zmysł słuchu? W tym rozdziale skupimy się na zastosowaniu Arduino do tworzenia dźwięków, byś mógł łatwiej odczytywać informacje ze swoich projektów.

Dźwięk za pomocą Arduino możesz generować na wiele sposobów. Najprostszą metodą jest użycie funkcji `tone()`, na której skupimy się w tym rozdziale najmocniej. Jednak możesz korzystać z różnych rozszerzeń (ang. *shield*), dzięki którym uzyskasz bardziej skomplikowane możliwości odtwarzania muzyki za pomocą Arduino i dodatkowego przetwarzania danych. (Rozszerzenia Arduino to oddzielne płytki zapewniające dodatkową funkcjonalność, które umieszczasz na górze swojego Arduino). Jeżeli masz Arduino Due, to do generowania dźwięków możesz wykorzystać wbudowany przetwornik cyfrowo-analogowy (DAC).

Zasada działania głośników

Zanim będziesz mógł tworzyć dźwięki za pomocą Arduino, musisz zrozumieć, czym jest dźwięk i jak jest odbierany przez człowieka. W pierwszej części tego podrozdziału dowiesz się, jak generowane są fale dźwiękowe, jakie są ich właściwości i jak przez manipulowanie nimi można uzyskać muzykę, głos itp.

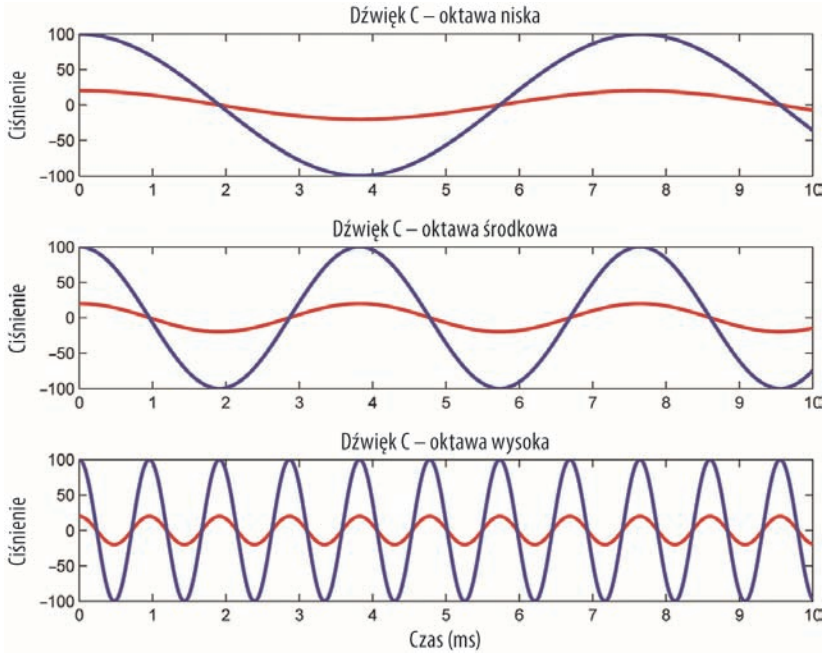
Właściwości dźwięku

Dźwięk jest przekazywany przez powietrze w postaci fali ciśnieniowej. Gdy obiekt taki jak głośnik, bęben czy dzwonek wibruje, wywołuje drgania powietrza wokół niego. Gdy cząsteczki powietrza drgają, przekazują energię do otaczających je cząsteczek, co powoduje, że te również drgają. W ten sposób fala ciśnieniowa jest przekazywana ze źródła do błony bębenkowej w uchu, przez wywołanie reakcji łańcuchowej drgających cząsteczek. Zatem dlaczego musisz to wiedzieć, aby zrozumieć, jak tworzyć dźwięki za pomocą Arduino?

Możesz sterować dwiema właściwościami tych drgających cząsteczek przy użyciu Arduino: częstotliwością i amplitudą. **Częstotliwość** określa, jak szybko cząsteczki powietrza drgają tam i z powrotem, a **amplituda** określa wielkość tych drgań. Z punktu widzenia fizyki dźwięki o większej amplitudzie są głośniejsze, a te o niższej — cichsze. Dźwięki o wysokiej częstotliwości są wyższe (jak sopran), a te o niższej mają niższy ton (jak bas). Przeanalizuj rysunek 6.1, na którym przedstawiono sinusoidalną falę dźwiękową o różnych częstotliwościach i amplitudach.

Na rysunku 6.1 przedstawiono dźwięk C w trzech oktawach: dolnej, środkowej i górnej. Każdy wykres pokazuje podane częstotliwości dla niskiej i wysokiej amplitudy. Na przykład, aby zrozumieć, czym jest częstotliwość i amplituda, skup się na dźwięku C oktawy środkowej. Częstotliwość tego dźwięku jest równa 261,63 herca (Hz). Innymi słowy, głośnik bądź struna gitary lub pianina wykona 261,63 drgania na sekundę, aby wytworzyć dźwięk C środkowej oktawy. Okres fali to odwrotność tej liczby, co łatwo zaobserwować na rysunku 6.1. Na wykresie długość jednego pełnego drgania jest równa 1/261,63, co daje 3,822 ms. Z wykorzystaniem Arduino możesz określić okres dla fali kwadratowej, a tym samym ton.

Co ważne, Arduino (poza Due, które ma przetwornik cyfrowo-analogowy z prawdziwego zdarzenia) nie potrafi stworzyć fali sinusoidalnej, którą możesz zaobserwować w prawdziwym świecie. Fala kwadratowa to cyfrowa fala okresowa, która również drga między stanem wysokim i niskim, ale tym razem zmiana jest niemal natychmiastowa, a nie powolna, jak w przypadku fali sinusoidalnej.



Rysunek 6.1. Fale dźwiękowe o różnych częstotliwościach i amplitudach (wykonano w programie MATLAB)

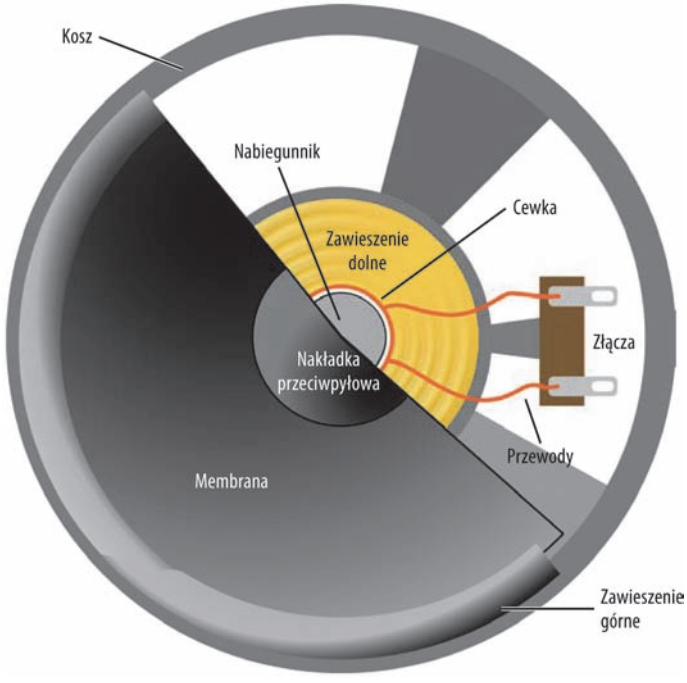
W dalszym ciągu powstaje fala ciśnieniowa, która skutkuje dźwiękiem, jednak dźwięk nie jest tak ładny jak w przypadku sinusoidy.

Jeżeli chodzi o amplitudę, to możesz nią sterować przez zmianę ilości prądu płynącego przez głośnik. Szeregowe połączenie potencjometru z głośnikiem pozwala dynamicznie zmieniać poziom głośności.

Zasada działania głośnika

Głośniki, podobnie do silników, o których była mowa w poprzednim rozdziale, korzystają z sił elektromagnetycznych, aby zamieniać elektryczność w ruch. Spróbuj przyłożyć metalowy przedmiot do tylnej części głośnika. Czy zauważyłeś coś ciekawego? Prawdopodobnie metalowy przedmiot przyczepił się do tylnej ścianki, gdyż każdy głośnik ma z tyłu spory magnes trwały. Rysunek 6.2 przedstawia przekrój najczęściej spotykanej konstrukcji głośnika.

Magnes trwały jest zamontowany za cewką i nabiegunnikiem, które są pokazane na rysunku. Podczas przesyłania sinusoidalnego sygnału napięcia (lub fali kwadratowej w przypadku Arduino) do przewodów cewki zmieniający się prąd wytwarza pole magnetyczne, które sprawia, że cewka i membrana drgają w górę i w dół, przyciągane przez magnes trwały, a następnie odpychane przez wytworzone pole magnetyczne. Ten ruch drgający tam i z powrotem wywołuje drganie powietrza z przodu głośnika, a co za tym idzie, tworzy się fala dźwiękowa, która może dotrzeć do błony bębenkowej w Twoim uchu.



Rysunek 6.2. Przekrój głośnika (źródło: Wikipedia, GNU FDL)

Wykorzystanie funkcji tone() do tworzenia dźwięków

Arduino IDE zawiera wbudowaną funkcję, która ułatwia tworzenie dźwięków z dowolną częstotliwością. Funkcja `tone()` generuje falę kwadratową o zadanej częstotliwości na wybranym przez Ciebie pinie wyjścia. Funkcja ta przyjmuje trzy argumenty, z czego ostatni jest opcjonalny:

- Pierwszym argumentem jest pin, na którym ma być wytworzony dźwięk.
- Drugi argument to częstotliwość dźwięku.
- Trzeci argument (nieobowiązkowy) określa długość dźwięku. Jeżeli trzeci argument zostanie pominięty, dźwięk będzie wybrzmiewał, dopóki nie wywołasz funkcji `noTone()`.

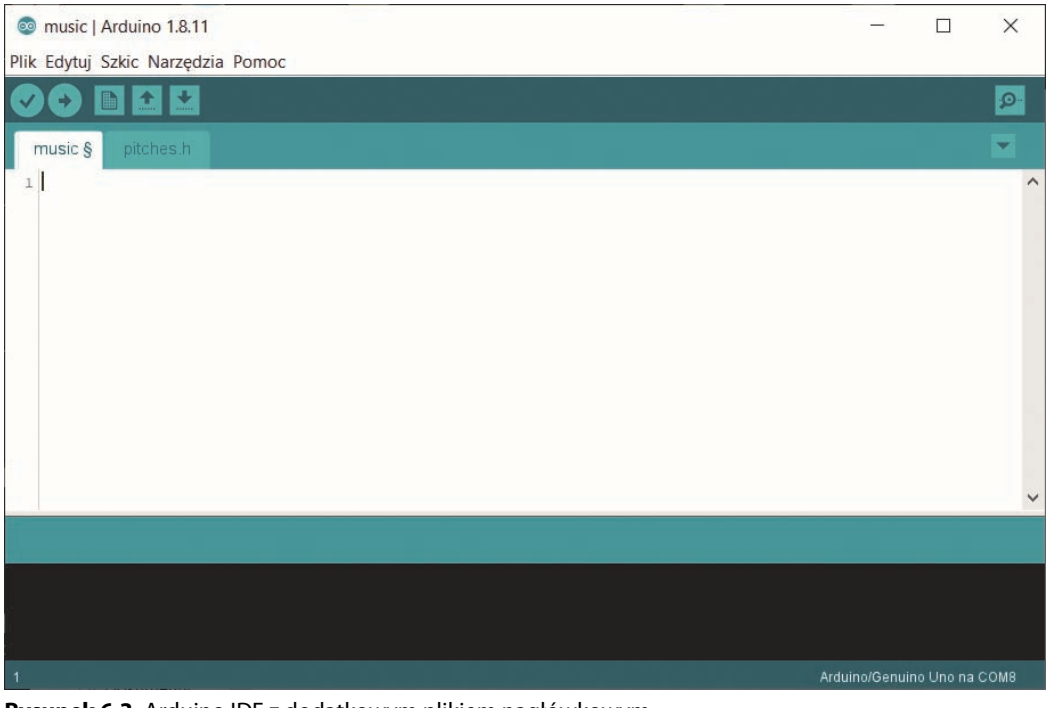
Ponieważ funkcja `tone()` wykorzystuje sprzętowy timer (układ liczący) ATmegi, dźwięk może być odtwarzany w tle, a Arduino może wykonywać inne zadania.

W następnych punktach nauczysz się, jak odtwarzać sekwencje złożone z losowych dźwięków. Gdy już to opanujesz, funkcji `tone()` możesz użyć do wydawania dźwięków w odpowiedzi na działanie różnych urządzeń wejścia (przyciski, czujniki odległości, akcelerometry itd.). Na końcu tego rozdziału zbudujesz proste, pięcioklawiszowe pianino, na którym będziesz mógł zagrać.

Dodanie własnego pliku nagłówkowego

W przypadku odtwarzania muzyki bardzo przydatny okazuje się plik przyporządkowujący częstotliwości odpowiednim nutom. Dzięki temu wygrywanie prostych melodyjek jest bardziej intuicyjne. Jeżeli czytanie nut nie jest Ci obce, to wiesz, że nuty są zapisywane między innymi za pomocą liter, które wskazują na wysokość dźwięku. Arduino IDE zawiera plik nagłówkowy, który z kolei zawiera informacje o zależnościach między nutami a ich częstotliwościami. Zamiast szukać tego pliku w folderze instalacyjnym Arduino, możesz go znaleźć w materiałach do pobrania, w folderze tego rozdziału. Umieścisz go w folderze swojego szkicu, gdy już go stworzysz.

Otwórz Arduino IDE i zapisz pusty szkic, który jest automatycznie tworzony, gdy otwierasz IDE. Jak już pewnie zauważyłeś, kiedy zapisujesz szkic, tworzy się folder o tej samej nazwie, gdzie umieszczany jest Twój plik z rozszerzeniem *.ino*. Umieszczając w tym folderze inne pliki, możesz dołączyć je do swojego programu, a jednocześnie zadbać o dobrą organizację kodu. Skopiuj plik *pitches.h* do folderu utworzonego przez IDE, a następnie zamknij Arduino IDE. Otwórz ponownie swój plik *.ino* w Arduino IDE i zauważ, że pojawiły się dwie zakładki (patrz rysunek 6.3).



Rysunek 6.3. Arduino IDE z dodatkowym plikiem nagłówkowym

Kliknij zakładkę *pitches.h*, aby zobaczyć zawartość pliku. Zauważ, że jest to po prostu lista łatwych do odczytania dla ludzi nazw nut, wraz z odpowiadającymi im częstotliwościami. Jednak samo umieszczenie pliku nagłówkowego w folderze nie wystarczy. Aby się upewnić, że kompilator skorzysta z danych umieszczonych w tym pliku, musisz, kompilując program na Arduino, wskazać mu ten plik. Jest to bardzo łatwe. Jedyne, co musisz zrobić, to dodanie poniższej linijki kodu na początku programu:

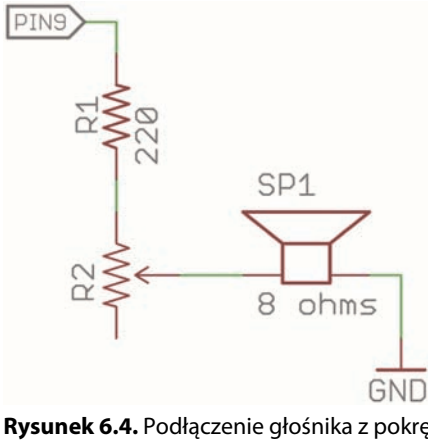
```
#include "pitches.h" // Plik nagłówkowy z definicją dźwięków
```

Dla kompilatora powyższa linijka kodu jest równoznaczna z wklejeniem zawartości pliku na początku pliku głównego. Jednak w ten sposób Twój szkic będzie bardziej uporządkowany i łatwiejszy do odczytania. W dalszych punktach napiszesz resztę kodu, gdzie tak naprawdę wykorzystasz definicje umieszczone w pliku *pitches.h*, który właśnie zaimportowałeś.

Podłączanie głośnika

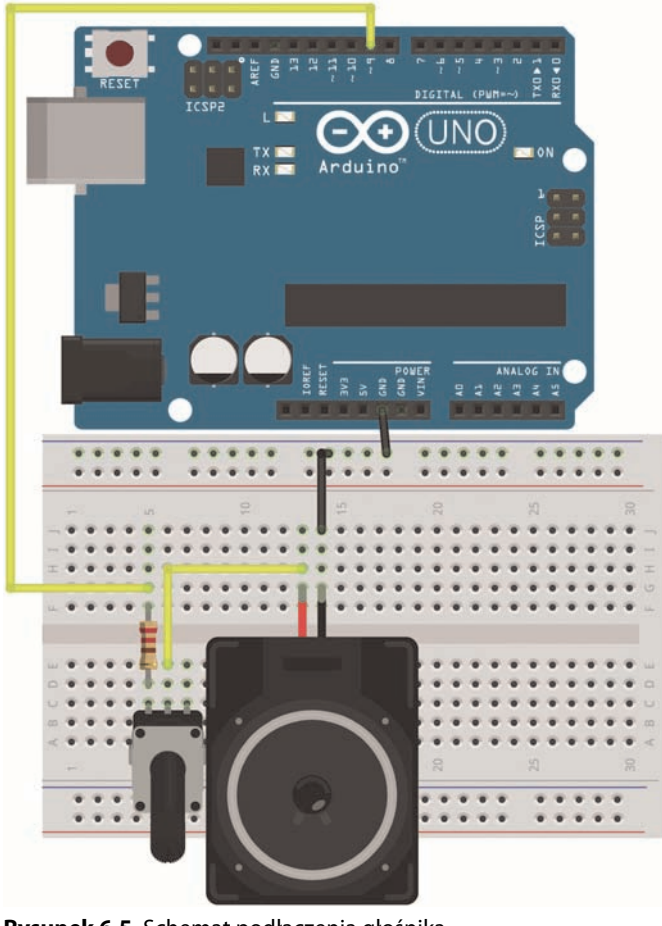
Gdy już dołączyłeś plik nagłówkowy z definicjami dźwięków, jesteś gotowy, aby zbudować obwód testowy i napisać prosty program, który będzie wygrywał melodię. Układ elektryczny jest dość prosty i wymaga tylko podłączenia głośnika do pinu wyjścia Arduino. Niemniej jednak pamiętaj o tym, czego nauczyłeś się w poprzednich rozdziałach o opornikach.

Tak jak w przypadku diod LED będziesz musiał połączyć szeregowo opornik i głośnik, aby się upewnić, że nie wyciągniesz za dużo prądu z jednego z pinów wejścia/wyjścia Arduino. Jak już się dowiedziałeś wcześniej, każdy z pinów I/O może dostarczyć maksymalnie tylko 40 mA, tak więc dobierz opornik, który zapobiegnie przekroczeniu tej wartości. Głośnik, który polecam, ma wewnętrzną rezystancję równą 8 Ω (tak jak większość dostępnych na rynku głośników), której źródłem są uzwojenia tworzące elektromagnes. Jeżeli wykorzystujesz głośnik o innej rezystancji, upewnij się, że odpowiednio zmodyfikowałeś tę wartość w następujących obliczeniach. Przypomnij sobie prawo Ohma, które mówi, że $U = I \cdot R$. W tym przypadku pin wejścia/wyjścia ma napięcie wyjściowe 5 V, a natężenie, którego nie chcesz przekroczyć, to 40 mA. Obliczywszy R, dowiesz się, że minimalna rezystancja, jaką musisz zastosować, równa się: $R = 5\ V : 40\ mA = 125\ \Omega$. Opór równy 8 Ω jest już zapewniony przez głośnik, zatem Twój opornik musi mieć wartość przynajmniej: 125 Ω – 8 Ω = 117 Ω. Powszechnie dostępnymi opornikami o zbliżonej wartości są oporniki 150 Ω i 220 Ω, więc możesz zastosować jeden z nich. Zastosowanie opornika 150 Ω będzie skutkowało trochę głośniejszym dźwiękiem niż w przypadku opornika 220 Ω, ale prawdopodobnie różnica będzie zbyt mała, abyś mógł ją zauważyć. Przez dobór rezystancji możesz zmieniać głośność przetwornika. Do tego celu wykorzystaj potencjometr połączony szeregowo z głośnikiem, tak jak zostało to pokazane na rysunku 6.4. Dzięki temu czynność dostosowywania głośności będzie bardzo prosta. Na schemacie *R1* to opornik 220 Ω, a *R2* to potencjometr.



Rysunek 6.4. Podłączenie głośnika z pokrętłem do regulacji poziomu głośności (wykonano w programie EAGLE)

Zauważ, że potencjometr jest podłączony w inny sposób, niż robiłeś to dotychczas. W tej konfiguracji wykorzystane są tylko dwa piny potencjometru: środkowy należy podłączyć do głośnika, a jeden ze skrajnych — do opornika 220 Ω. W momencie gdy potencjometr zostaje maksymalnie przekręcony w stronę niepodłączonego wyprowadzenia, cała jego rezystancja jest dodawana do rezystancji opornika 220 Ω podłączonego szeregowo, a poziom głośności się zmniejsza. Gdy pokrętło potencjometru jest przekręcone maksymalnie w stronę podłączonego pinu, do opornika podłączonego szeregowo nie jest dodawana żadna rezystancja, a poziom głośności jest ustawiony na maksimum. Patrząc na rysunek 6.4, podłącz swój głośnik do Arduino. Następnie, porównując swoją pracę z rysunkiem 6.5, upewnij się, że wszystko dobrze podłączyłeś. Jeżeli Twój głośnik nie ma gotowych przewodów dołączonych do jego złączy, możesz je przylutować. Jeżeli nie masz pod ręką lutownicy, możesz dokładnie i ciasno owinąć druciki przewodu wokół oczka złącza. Chociaż ta metoda zadziała, zalecane jest przylutowanie przewodów.



Rysunek 6.5. Schemat podłączenia głośnika (wykonano w programie Fritzing)

Głośniki nie są spolaryzowane, więc możesz je podłączyć w dowolnym kierunku. Po udanym podłączeniu głośnika jesteś gotowy, aby tworzyć muzykę!

Tworzenie sekwencji dźwięków

Przed odtwarzaniem jakichkolwiek melodii nauczysz się wykorzystywać tablice do łatwego przechowywania wielu wartości. Następnie zastosujesz prostą pętlę, która przejdzie przez kolejne nuty zapisane w tablicy i odtworzy je za pomocą głośnika.

Stosowanie tablic

Tablica jest zestawem danych, które są ze sobą w pewien sposób powiązane. Zgrupowane wartości doskonale nadają się do odczytywania ich po kolei. Możesz sobie wyobrazić, że tablica to ponumerowana lista. Każda pozycja ma swój indeks, który wskazuje na położenie na liście, a do każdego indeksu przypisana jest wartość, którą chcesz przechować. W tym przykładzie wykorzystasz tablicę do przechowania nut w takiej kolejności, w jakiej mają być odtworzone.

Aby mieć pewność, że pamięć Arduino jest w odpowiedni sposób zarządzana, musisz zadbać o to, by zadeklarowane tablice miały określoną długość. W tym celu możesz podać wprost liczbę elementów lub od razu wypisać wszystkie elementy, które chcesz zapisać w tablicy. I tak na przykład, jeśli chcesz utworzyć tablicę, która będzie przechowywać cztery liczby całkowite, możesz to zrobić w następujący sposób:

```
int numbers[4];
```

Podczas deklarowania tablicy możesz od razu podać jej elementy. Gdy podajesz wartości, nie musisz określać długości tablicy w nawiasach kwadratowych. Jeżeli długość tablicy nie jest podana, kompilator zakłada, że jest równa liczbie elementów.

```
// Dopuszczalne są obie wersje
int numbers[4] = {-7, 0, 6, 234};
int numbers[] = {-7, 0, 6, 234};
```

Zauważ, że numeracja indeksów tablicy zaczyna się od zera. Innym słowy, pierwsza liczba jest wpisana na pozycję 0, druga na pozycję 1 itd. Do elementów tablicy odnosimy się przez podanie nazwy tablicy i indeksu elementu w nawiasie kwadratowym. Na przykład, jeżeli chcesz ustawić jasność diody LED podłączonej do pinu 9 na wartość, która jest trzecim elementem tablicy, możesz to zrobić w następujący sposób:

```
analogWrite(9,numbers[2]);
```

Zwróć uwagę, że ponieważ indeksowanie zaczyna się od zera, trzeci element tablicy ma indeks 2. Jeżeli chcesz zmienić wartość elementu tablicy, możesz to zrobić w podobny sposób:

```
numbers[2] = 10;
```

W następnym kroku wykorzystasz tablice (w taki sam sposób jak w powyższych przykładach), aby stworzyć strukturę, która może przechowywać sekwencję nut do odtworzenia na głośniku.

Tworzenie tablic z nutami i ich długością

Możesz wykorzystać dwie tablice o tej samej długości, aby zapisać informację o melodii, którą chcesz odegrać. Pierwsza tablica będzie zawierać listę dźwięków, a elementy drugiej będą czasem trwania każdej nuty w milisekundach. Następnie możesz odczytywać kolejno elementy każdej z tablic przez odwoływanie się do ich indeksów i odegrać swoją melodyjkę.

Z wykorzystaniem swoich ubogich umiejętności muzycznych, które zdobyłem na lekcjach muzyki w szkole średniej, złożyłem krótką i wpadającą w ucho melodyjkę.

```
// Tablica z nutami
int notes[] = {
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_E4, NOTE_D4, NOTE_C4, NOTE_B4, NOTE_A4, NOTE_B4, NOTE_C4, NOTE_D4,
 NOTE_E4, NOTE_E3, NOTE_A4, 0
};

// Czas trwania każdej nuty (w ms)
int times[] = {
 250, 250, 250, 250,
 250, 250, 250, 250,
 125, 125, 125, 125, 125, 125, 125, 125,
 250, 250, 250, 250
};
```

Zauważ, że obie tablice mają taką samą długość: 20 elementów. Zwróć też uwagę na to, że niektóre nuty są oznaczone jako 0. To są pauzy (cisza w muzyce). Każda nuta ma swoją parę z tablicy z długościami trwania każdej z nich. Jeżeli znasz się na muzyce, to zauważysz, że długość trwania ćwierćnuty ustawiłem na 250 ms, a ósemki na 125 ms. Melodia ma metrum 4/4.

Najpierw wypróbuj moją melodię, a następnie napisz własną!

Składanie programu w całość

Ostatni krok polega na dodaniu do szkicu kodu odpowiedzialnego za odtworzenie utworu muzycznego. Można to zrobić z użyciem prostej pętli `for`, która przejdzie przez wszystkie indeksy tablic i odtworzy nuty z odpowiednią długością. Ponieważ prawdopodobnie nie będziesz chciał słuchać tej melodii w kółko, kod, który ją odtwarza, umieść w funkcji `setup()`, aby melodia wybrzmiała tylko raz. Możesz ponownie odegrać melodię przez naciśnięcie przycisku *Reset*. Listing 6.1 przedstawia kompletny program.

Listing 6.1. Odtwarzacz muzyki na Arduino — music.ino

```
// Odtwarzanie melodii za pomocą głośnika

#include "pitches.h" // Plik nagłówkowy z definicją dźwięków

const int SPEAKER = 9;  // Pin, do którego podłączony jest głośnik

// Tablica z nutami
int notes[] = {
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_A4, NOTE_E3, NOTE_A4, 0,
 NOTE_E4, NOTE_D4, NOTE_C4, NOTE_B4, NOTE_A4, NOTE_B4, NOTE_C4, NOTE_D4,
 NOTE_E4, NOTE_E3, NOTE_A4, 0
};

// Czas trwania każdej nuty (w ms)
int times[] = {
 250, 250, 250, 250,
 250, 250, 250, 250,
 125, 125, 125, 125, 125, 125, 125, 125,
 250, 250, 250, 250
```

```
};


void setup()
{
 // Zagraj każdą nutę odpowiednio długo
 for (int i = 0; i < 20; i++)
 {
  tone(SPEAKER, notes[i], times[i]);
  delay(times[i]);
 }
}

void loop()
{
 // Naciśnij przycisk Reset, aby ponownie odtworzyć melodię.
}
```

Jeżeli chcesz stworzyć własny utwór muzyczny, upewnij się, że obie tablice są tej samej długości i że odpowiednio dostosowałeś warunek pętli `for`. Ponieważ funkcja `tone()` może działać w tle, ważne jest, by użyć funkcji `delay()`. Opóźniwszy program o długość wygrywanej nuty, będziesz miał pewność, że Arduino nie zacznie odtwarzać następnej, zanim ta nie wybrzmi do końca.

Ograniczenia funkcji tone()

Funkcja `tone()` ma kilka ograniczeń, o których musisz wiedzieć. Podobnie jak biblioteka obsługująca serwomechanizmy funkcja `tone()` polega na timerze, który również jest wykorzystywany przez modulację szerokości impulsów (PWM) płytki. Jeżeli korzystasz z `tone()`, PWM nie będzie działać poprawnie na pinach 3 i 11 (na wszystkich płytkach Arduino poza Mega).

Pamiętaj też, że piny wejścia/wyjścia Arduino nie są przetwornikami cyfrowo-analogowymi (DAC). Zatem na ich wyjściu uzyskasz tylko falę kwadratową o podanej częstotliwości, a nie falę sinusoidalną. Choć wystarczy to do odgrywania dźwięków za pomocą głośnika, możesz uznać, że nie jest to rozwiązanie wystarczająco dobre do odtwarzania muzyki. Jeżeli chcesz odtwarzać pliki WAV, możesz użyć rozszerzeń (takich jak Adafruit Wave Shield lub SparkFun MP3), zastosować zewnętrzny przetwornik cyfrowo-analogowy lub wykorzystać ten wbudowany w Arduino Due i przeznaczoną do tego modelu bibliotekę audio.

Ostatnim ograniczeniem funkcji `tone()` jest to, że w danym momencie możesz ją stosować tylko na jednym pinie, więc nie jest najlepszym wyborem, jeżeli chcesz użyć więcej niż jednego głośnika. Jeśli chcesz użyć kilku w jednym czasie za pomocą standardowego Arduino, musisz zastosować manualny licznik do sterowania przerwaniami — coś, o czym dowiesz się więcej w rozdziale 13., zatytułowanym „Przerwania i inne funkcje specjalne”.

Budowa mikropianina

Dodanie melodyjek jako sygnałów informacyjnych dla własnych projektów jest świetnym pomysłem. Na przykład rozważ zastąpienie diody LED dźwiękiem lub dodanie do niej dźwięku, który potwierdzałby daną akcję. Ale co w przypadku, gdybyś chciał dynamicznie sterować dźwiękiem? Pod koniec

tego rozdziału zbudujesz proste pentatoniczne pianino. Skala pentatoniczna składa się tylko z pięciu dźwięków na oktawę, a nie, jak zazwyczaj, z siedmiu. Co ciekawe, w skali pentatonicznej występuje tylko minimalny dysonans między dźwiękami, co oznacza, że dźwięki te zawsze brzmią ze sobą dobrze, Zatem wykorzystanie nut ze skali pentatonicznej do stworzenia prostego pianina ma sens.

Uwaga **Rękawica SudoGlove to jedno z urządzeń, za pomocą których możesz syntetyzować dźwięki skali pentatonicznej. Więcej informacji o tym produkcie (w języku angielskim) znajdziesz na stronie *sudoglove.com*.**

W swoim pianinie Arduino zastosujesz następujące dźwięki skali pentatonicznej: C, D, E, A, G. Co do oktawy: możesz wybrać dowolną, która Ci się najbardziej podoba. Ja wybrałem czwartą oktawę z pliku nagłówkowego.

Na początku podłącz pięć przycisków do swojego Arduino. Tak jak w przypadku przycisków omówionych w rozdziale 2., zatytułowanym „Cyfrowe wejścia, wyjścia i modulacja szerokości impulsów (PWM)", musisz do każdego z nich zastosować opornik 10 kΩ. W tym przykładzie nie będziesz musiał martwić się problemem drgających styków, gdyż dźwięk będzie odtwarzany tylko w momencie, gdy przycisk będzie wciśnięty. Podłącz przyciski tak, jak zostało to pokazane na rysunku 6.6, a głośnik pozostaw podłączony w ten sam sposób co wcześniej.

Kod dla pianina jest tak naprawdę bardzo prosty. Podczas każdego obiegu pętli sprawdzany jest każdy przycisk. W rezultacie nuta wybrzmiewa tak długo, jak przycisk jest wciśnięty. W tym programie funkcja `tone()` jest wywoływana bez argumentu określającego długość trwania dźwięku, gdyż będzie on trwał dopóty, dopóki przycisk będzie wciśnięty. Aby z głośnika nie wydobywały się dźwięki, gdy żaden przycisk nie jest wciśnięty, na końcu pętli głównej wywoływana jest funkcja `noTone()`. Ponieważ potrzebnych jest tylko kilka nut, możesz je skopiować z pliku nagłówkowego i wkleić bezpośrednio do głównego pliku programu. Do nowego szkicu wklej kod z listingu 6.2 i wyślij go do Arduino. Teraz możesz już grać improwizowane melodie na swoim pianinie!

Listing 6.2. Pentatoniczne mikropianino — piano.ino

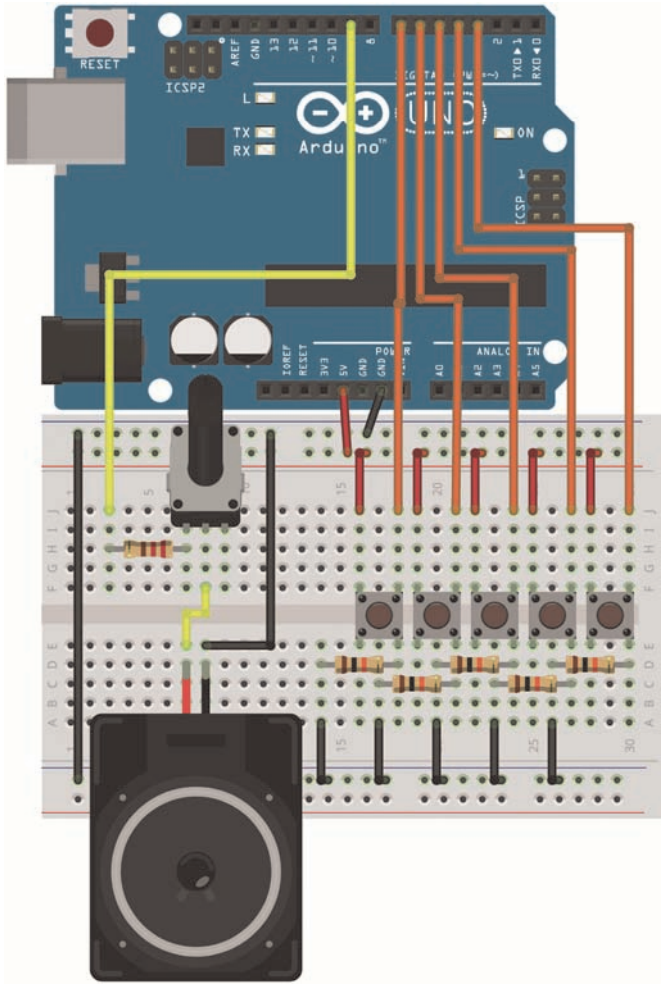
```
// Pentatoniczne pianino
// C D E G A

#define NOTE_C  262 // Hz
#define NOTE_D  294 // Hz
#define NOTE_E  330 // Hz
#define NOTE_G  392 // Hz
#define NOTE_A  440 // Hz

const int SPEAKER = 9;    // Głośnik podłączony do pinu 9

const int BUTTON_C = 7;  // Pin, do którego podłączony jest przycisk
const int BUTTON_D = 6;  // Pin, do którego podłączony jest przycisk
const int BUTTON_E = 5;  // Pin, do którego podłączony jest przycisk
const int BUTTON_G = 4;  // Pin, do którego podłączony jest przycisk
const int BUTTON_A = 3;  // Pin, do którego podłączony jest przycisk


void setup()
{
 // Niewymagana jest żadna konfiguracja
```



Rysunek 6.6. Schemat podłączeniowy mikropianina (wykonano w programie Fritzing)

```
// Funkcja tone() ustawia piny jako wyjście
}

void loop()
{
 while (digitalRead(BUTTON_C))
  tone(SPEAKER, NOTE_C);
 while(digitalRead(BUTTON_D))
  tone(SPEAKER, NOTE_D);
 while(digitalRead(BUTTON_E))
  tone(SPEAKER, NOTE_E);
 while(digitalRead(BUTTON_G))
  tone(SPEAKER, NOTE_G);
 while(digitalRead(BUTTON_A))
  tone(SPEAKER, NOTE_A);
```

```
  // Wycisz wszystkie dźwięki, gdy żaden przycisk nie jest naciśnięty
  noTone(SPEAKER);
}
```

Każda pętla `while()` będzie ciągle wywoływać funkcję `tone()` z odpowiednią częstotliwością dopóty, dopóki przycisk jest wciśnięty. Stan przycisku jest odczytywany w warunku pętli `while()`, aby nie trzeba było uprzednio zapisywać odczytu jako tymczasowej wartości zmiennej. Funkcja `digitalRead()` zwraca wartość typu *bool* równą „prawda”, gdy tylko wejście przycisku ma stan wysoki, a zwracana wartość może być odczytywana bezpośrednio przez pętlę `while()`. Aby Twój kod był bardziej uporządkowany, możesz pominąć nawiasy klamrowe, gdy zawartość pętli to tylko jedna linia, tak jak w tym przykładzie. Jeżeli zawartość pętli składa się z kilku linijek, musisz umieścić je w nawiasach klamrowych, tak jak to robiłeś w poprzednich przykładach.

Podsumowanie

W tym rozdziale nauczyłeś się, że:

- głośniki generują falę ciśnieniową, która rozchodzi się w powietrzu i jest odbierana przez Twoje uszy jako dźwięk,
- zmieniający się prąd elektryczny powoduje powstanie pola magnetycznego, które może być wykorzystane do wytworzenia dźwięku w głośniku,
- funkcja `tone()` może być wykorzystana do wytworzenia dźwięków o dowolnej częstotliwości i długości,
- język programowania Arduino zawiera tablice danych, a przez jej poszczególne elementy można przechodzić za pomocą pętli,
- głośność przetwornika można regulować za pomocą potencjometru połączonego z nim szeregowo.

ARDUINO: OTO ŚWIAT ŁĄCZĄCY INŻYNIERIĘ I MAGIĘ!

Mikrokontroler Arduino szybko może stać się Twoim ulubionym narzędziem, pasją albo wstępem do fascynującego świata elektroniki, programowania, systemów sterujących, interakcji człowieka z komputerem, a nawet... sztuki! Arduino może zrobić wszystko, co tylko zechcesz: sterować domową uprawą ziół, być serwerem sieciowym albo autopilotem czterośmigłowego drona. Umożliwia korzystanie z intuicyjnego języka programowania, a także pozwala się łatwo doposażyć w czujniki, serwomechanizmy, oświetlenie, głośniki, dodatkowe moduły i inne układy scalone. Trudno wymienić wszystkie możliwości Arduino, gdyż ogranicza je tylko wyobraźnia.

Oto zaktualizowane i rozszerzone wydanie książki, dzięki której poznasz wybrane zagadnienia z dziedziny fizyki, elektroniki i cyfrowego projektowania. Omówiono też podstawy algorytmów i charakterystyczne dla Arduino koncepcje programistyczne. Znalazły się tu zaktualizowane projekty i nowe tematy, takie jak łączność bezprzewodowa czy silniki krokowe, oraz dużo więcej wiadomości związanych z elektrotechniką i projektowaniem produktu. Dowiesz się, jak łączyć ze sobą różne elementy, a także jak czytać schematy i w jaki sposób dobierać odpowiednią część dla konkretnego projektu. Nauczysz się projektować i tworzyć kompletny kod. Przekonasz się, że przedstawione tu praktyki z zakresu elektrotechniki, projektowania systemów i programowania można szeroko stosować, również poza światem Arduino.

W książce:

- funkcjonalność płytek Arduino i ich wykorzystywanie do różnych zadań
- czujniki cyfrowe, analogowe i interfejsy komunikacyjne
- zasady projektowania systemu, programowania i elektrotechniki
- fragmenty kodu, najlepsze praktyki oraz gotowe do zastosowania schematy systemów
- budowa elementów, które można podłączyć do internetu

Jeremy Blum od lat zajmuje się skomplikowaną elektromechaniką. Zaprojektował obrotnicę solarną, protezy ręki, roboty poruszające się po kratownicy, thereminy (instrumenty muzyczne), kontrolery gestów, systemy automatyki domowej, urządzenia wykorzystujące rzeczywistość rozszerzoną i wiele innych. Jest znany z edukacyjnych filmów umieszczonych w serwisie YouTube. Wcześniej pracował jako inżynier sprzętu w Google, obecnie jest dyrektorem technicznym w firmie Shaper Tools.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-6923-8
9 788328 369238
Cena: 89,00 zł
INFORMATYKA W NAJLEPSZYM WYDANIU
WILEY