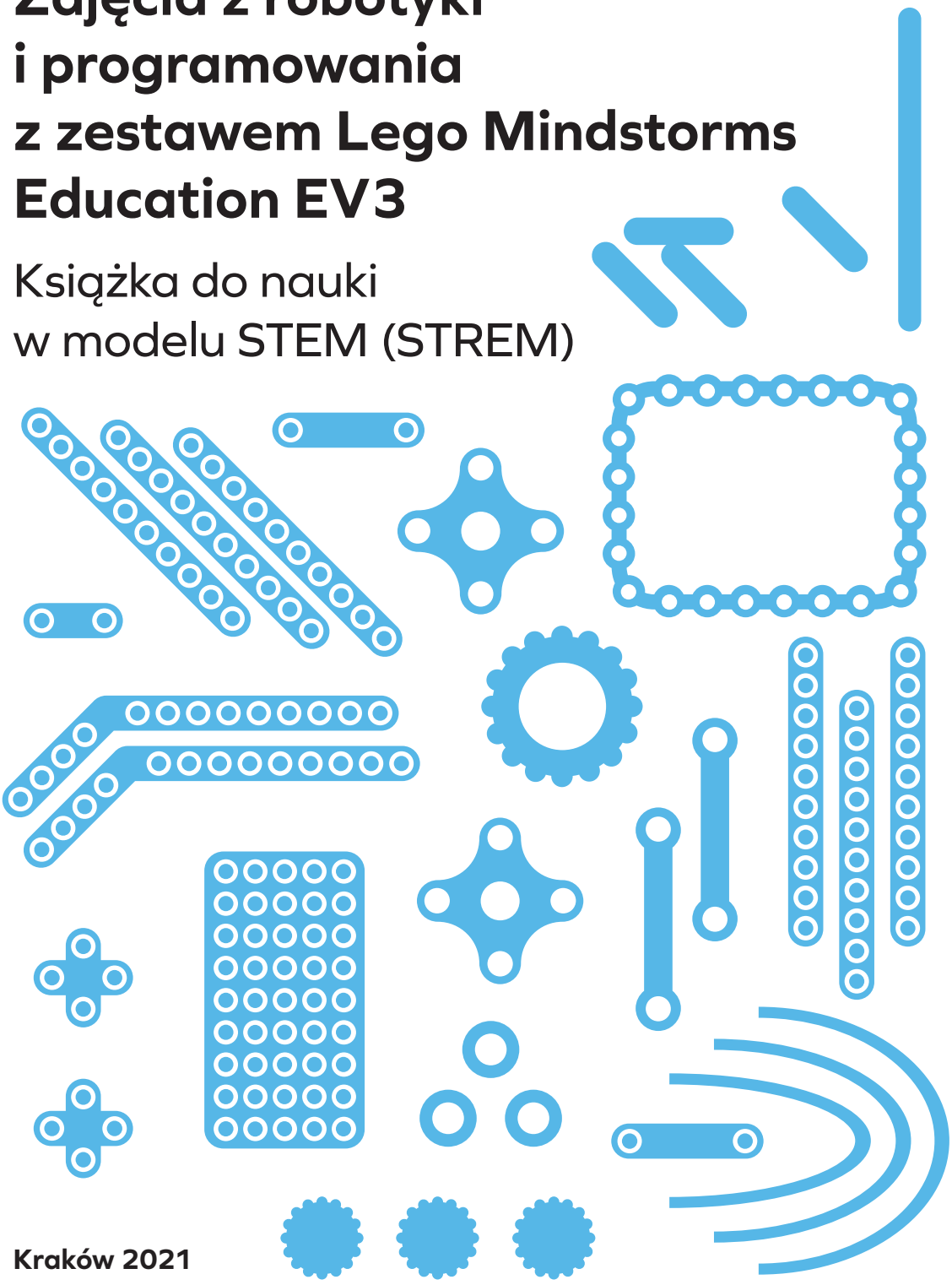
Radosław Antoni Kycia

Politechnika Krakowska
im. Tadeusza Kościuszki

Zajęcia z robotyki i programowania z zestawem Lego Mindstorms Education EV3

Książka do nauki w modelu STEM (STREM)



Kraków 2021

PRZEWODNICZĄCY KOLEGIUM REDAKCYJNEGO WYDAWNICTWA POLITECHNIKI KRAKOWSKIEJ
Tadeusz Tatara

PRZEWODNICZĄCY KOLEGIUM REDAKCYJNEGO WYDAWNICTW DYDAKTYCZNYCH
Elżbieta Węcławowicz-Bilska

REDAKTOR SERII
Wydział Inżynierii Materiałowej i Fizyki
Ludwik Byszewski

RECENZENT
Piotr Artiemjew

SEKRETARZ SEKCJI
Agnieszka Filosek

OPRACOWANIE REDAKCYJNE
Małgorzata Sikora

SKŁAD I ŁAMANIE
Anna Pawlik

PROJEKT OKŁADKI
Karolina Szafran

© Copyright by Politechnika Krakowska

ISBN 978-83-66531-81-9

Wydawnictwo PK, ul. Skarżyńskiego 1, 31-866 Kraków; tel. 12 628 37 25, fax 12 628 37 60
e-mail: wydawnictwo@pk.edu.pl ☐ www.wydawnictwo.pk.edu.pl
Adres do korespondencji: ul. Warszawska 24, 31-155 Kraków

Druk i oprawę wykonano w Dziale Poligrafii Politechniki Krakowskiej.
Ark. wyd. 4,5

Zam. 75/2021 Nakład 90 egz. Cena 30,45 zł z VAT

Kobietom mojego życia

Spis treści

1. Uwagi początkowe 7
   1.1. STEM 7
   1.2. Materiały dodatkowe 8
   1.3. Podziękowania 8
2. Przygotowanie 9
   2.1. Wstęp 9
   2.2. Przygotowanie zestawu 11
3. Mechanika 14
   3.1. Przekładnie 14
      3.1.1. Mechanizm różnicowy 16
   3.2. Śruba 17
   3.3. Zamiana ruchu obrotowego na inne formy 17
      3.3.1. Mimośród 18
   3.4. Krzywki 18
   3.5. Podsumowanie 19
4. Programowanie 20
   4.1. Teoria 20
   4.2. Przygotowanie 23
   4.3. Opis języka programowania 25
      4.3.1. Zielony 25
      4.3.2. Pomarańczowy 25
      4.3.3. Żółty 26
      4.3.4. Czerwony 26
      4.3.5. Niebieski 28
   4.4. Podsumowanie 28
5. Czujniki i silniki 29
   5.1. Opis silników 29
   5.2. Opis czujników 29
      5.2.1. Ultradźwiękowy czujnik odległości 29
      5.2.2. Czujnik kolorów 31
      5.2.3. Czujnik dotykowy 31
      5.2.4. Żyroskop 31
   5.3. Podsumowanie 33

6. Robotyka ... 34
6.1. Robot na kołach ... 34
6.1.1. Platforma na kołach ... 34
6.1.2. Platforma na kołach wykrywająca przeszkody ... 35
6.1.3. Platforma na kołach - podążamy za grubą linią ... 37
6.2. Robot kroczący ... 39
6.2.1. Budowa platformy kroczącej ... 40
6.2.2. Wykrywanie przeszkód ... 40
6.3. Mechanika ręki ... 41
6.4. Podsumowanie ... 42
7. Zakończenie ... 43
Dodatek A. Przygotowanie zajęć – dla opiekunów ... 47
Dodatek B. Mechanika ... 48
B.1. Skalar i wektor ... 48
B.1.1. Wielkość skalarna ... 48
B.1.2. Wektor ... 49
B.2. Zasady dynamiki Newtona ... 52
B.3. Maszyny proste ... 58
B.3.1. Dźwignia ... 58
B.3.2. Przekładnia ... 59
B.3.3. Śruba ... 60
B.4. Ogólne spojrzenie na konstrukcję układów mechanicznych ... 61
Dodatek C. Instalacja oprogramowania ... 63
C.1. PC z systemem Windows, połączenie przez Bluetooth – zalecane ... 63
C.2. PC z systemem Windows, połączenie przez kabel USB ... 64
C.3. PC z systemem Linux ... 65
C.4. Smartfon/tablet ... 65
Dodatek D. Praca w grupie ... 66
D.1. Inżynieria oprogramowania ... 66
D.2. Scrum ... 66
D.3. Symulacja ... 67
Pytania kontrolne ... 69
Bibliografia ... 71

Rozdział 1

Uwagi początkowe

Książka ta jest skryptem do zajęć z robotyki z zestawem LEGO® Mindstorms® EV3 dla młodzieży. Poszczególne części publikacji skierowane są do uczniów szkół ponadpodstawowych oraz instruktorów prowadzących zajęcia. Proponowane przykłady lekcji mają pobudzać młodzież do przyswajania wiedzy z dziedziny nauk ścisłych (informatyka, matematyka, fizyka), a także do kreatywnego rozwiązywania problemów.

Autor jednak nie odpowiada za szkody wynikające z realizacji materiału zawartego w tej książce. Dołożono wszelkich starań, aby materiał opisany był sprawdzony, a także nie powodował żadnych poważnych problemów. Jednak, jak to ma zawsze miejsce, gdy tworzymy ruchome lub autonomicznie poruszające się mechanizmy, istnieje ryzyko wyrządzenia szkody lub uszkodzenia ciała. Dlatego podczas zajęć osoba nadzorująca powinna przewidzieć skutki działań swoich podopiecznych i uchronić młodzież przed ewentualnymi uszkodzeniami ciała lub szkodami wyrządzonymi przez konstruowane mechanizmy.

Wszelkie znaki towarowe, jak LEGO® Mindstorms® EV3, USB, Microsoft Windows®, oraz inne, zostały użyte jedynie w celach informacyjnych. Książka nie jest oficjalnie firmowana przez żadną z firm, do których te nazwy należą.

Wszelkie nazwy, ilustracje i cytaty zostały użyte w książce do celów naukowych, omówienia i dyskusji dozwolonych w ramach prawa cytatu. Ilustracje, które nie zawierają w swoim opisie źródła, są pracami autora.

1.1. STEM

STEM, ang. *Science, Technology, Engineering, and Mathematics* (nauki ścisłe, technologia, inżynieria, matematyka) jest ideą nauki łączącą wszystkie wymienione dziedziny w jedną całość w myśl naturalnego założenia, że dziedziny te przenikają się. Ma to miejsce szczególnie w przypadku robotyki, do której ta książka jest elementarnym wprowadzeniem. Skupiono się więc na przekazywaniu wiedzy zgodnie z ideą STREM, ang. *Science, Technology, Robotics, Engineering, and Mathematics* (nauki ścisłe, technologia, robotyka, inżynieria, matematyka).

1.2. Materiały dodatkowe

Materiały dodatkowe do zajęć, zdjęcia i filmy znajdują się na stronie http://fizyk.ifpk.pk.edu.pl/~rkycia/roboty/roboty.html.

Ilustracja 1.1. Kod QR do strony z materiałami dodatkowymi

1.3. Podziękowania

Materiał powstał na podstawie zajęć prowadzonych dla uczniów szkół ponadpodstawowych na Politechnice Krakowskiej w ramach programu „3-maj z fizyką – trzecia misja uczelni" POWR.03.01.00-00-T021/18-00.

Dziękuję uczestnikom zajęć za zaangażowanie i pasję do odkrywania nieznanego.

Książka została sfinansowana ze środków Wydziału Inżynierii Materiałowej i Fizyki Politechniki Krakowskiej.

Rozdział 2

Przygotowanie

2.1. Wstęp

Lego® Mindstorms® EV3 w wersji edukacyjnej umożliwia konstrukcje robotów – od prostych do zaawansowanych. W książce omówiono następujące zagadnienia:

- Konstrukcje podstawowych mechanizmów oraz fizykę, a także matematykę potrzebną do ich opisu – mechanikę robotów;
- Podstawy programowania robotów – programowanie;
- Zasady działania czujników oraz ich wykorzystanie – sensory;
- Konstrukcje prostych robotów – robotykę.
- Na wstępie należy wspomnieć, że robotyka jest multidyscyplinarną nauką zajmującą się konstrukcją szeroko rozumianych robotów. Dzięki tej książce poznasz następujące dziedziny:
- Fizykę i matematykę prostych konstrukcji mechanicznych;
- Programowanie – będziesz musiał/a napisać program, aby powiedzieć robotowi, co ma zrobić;
- Mechatronikę – dowiesz się, jak połączyć elementy mechaniczne z elektronicznymi;
- Robotykę – nauczysz się konstruować roboty, które wykonują konkretne czynności określone przez program, który napiszesz. Nauczysz się tez używać sensorów, aby dać robotom możliwość „odczuwania" otaczającego ich świata oraz podejmowania decyzji na podstawie zebranych informacji.

Nie przejmuj się tymi specjalistycznymi terminami, gdyż wszystkiego dowiesz się w trakcie zabawy, którą już wkrótce zaczniemy!

Na początku jednak trzeba przypomnieć kilka zasad bezpieczeństwa:

- Utrzymuj porządek i organizuj miejsce pracy – zestaw Lego zawiera bardzo dużo elementów, dlatego powinieneś/aś zachować porządek na stanowisku pracy i starać się odkładać wszystkie rzecz na właściwe miejsca, aby klocki nie zgubiły się i aby zapanować nad chaosem w miejscu pracy.
- Zachowaj bezpieczeństwo – klocki zawierają małe elementy, których absolutnie nie można połykać. Po-

nadto klocki zawierają części ruchome, więc podczas ich działania należy zachować ostrożność i starać się przewidzieć skutek ich działania. Najbezpieczniej jest włączyć skonstruowany mechanizm i od razu odsunąć się na bezpieczną odległość, podziwiając efekt jego działania. Nie należy wkładać palców w ruchome mechanizmy, gdyż grozi to ich przytrzaśnięciem lub stłuczeniem.

- Uważaj na części elektryczne – klocki zasilane są zestawem baterii lub baterią o niskim napięciu. Napięcie prądu jest bezpieczne dla człowieka, jednak staraj się zachować ostrożność, gdy masz podejrzenie, że możesz mieć kontakt z przewodem. Jeżeli zauważysz, ze przewód jest uszkodzony, przerwij pracę i powiadom prowadzącego zajęcia.
- Szanuj zestaw i miejsce pracy – klocki służą do nauki i zabawy dla wielu grup i po twoich zajęciach będą wykorzystywane przez innych. Staraj się ich nie uszkodzić i nie zgubić, aby kolejni uczniowie mogli z nich korzystać.
- Organizuj miejsce pracy tak, aby umieścić pojemnik z częściami możliwie daleko od obszaru roboczego, aby zapobiec przypadkowemu trąceniu go i rozsypaniu części. Jeżeli taki przypadek się wydarzy, przerwij zajęcie i posprzątaj stanowisko przed wznowieniem pracy. Upewnij się, czy posprzątałeś wszystko.
- Nie bój się zapytać – prowadzący zajęcia opiekują się tobą i są po to, aby zapewnić ci maksimum korzyści wynikających z nauki robotyki i programowania. Jeżeli czegoś nie zrozumiałeś/aś lub masz wątpliwości, nie bój się zapytać – oni są dla ciebie.
- Pozostaw miejsce pracy w porządku po zakończeniu zajęć.

Poza zasadami, których trzeba przestrzegać, warto zaznaczyć, że:

- Na zajęciach masz praktycznie nieograniczoną możliwość eksperymentowania i realizowania swoich pomysłów. Jeżeli potrzebujesz pomocy, nie bój się porozmawiać z prowadzącymi zajęcia.
- Pracujemy w grupach, aby rozwijać zdolności miękkie/interpersonalne. Ułatwiają one funkcjonowanie w społeczeństwie i efektywną pracę w grupach. Zdolności te są obecnie wymagane podczas rekrutacji w różnych miejscach pracy, dlatego postaraj się jak najwięcej wynieść z naszych zajęć. Odnoś się do innych z szacunkiem. Jeżeli twoja grupa ma problemy z organizacją i współpracą, powinniście spróbować wdrożyć efektywne metodyki pracy w grupie. Jedna z nich opisana jest w załączniku D. Poświęć chwilę czasu, aby przeanalizować opisaną tam symulację pracy w grupie.

Na koniec zapamiętaj hasło, które przypisuje się amerykańskiemu komikowi Willowi Farrellowi: „Jeżeli nie przybył nikt z przyszłości, aby Cię powstrzy-

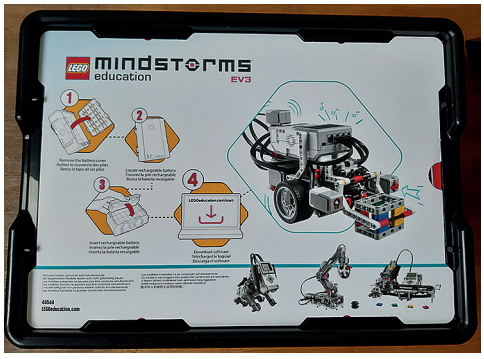
mać, to jak złe mogą być Twoje decyzje?" (ang. *If nobody comes from the future to stop you, how bad can the decision really be?*).

2.2. Przygotowanie zestawu

Jeżeli jesteś instruktorem, przed przystąpieniem do zajęć przeczytaj Załącznik A.

Weź do rąk pudełko z zestawem Lego® Mindstorms® EV3 (ilustracja 2.1). Zestaw zawiera wiele elementów, dlatego należy zachować ostrożność, aby ich nie zgubić. Na początku sprawdź, czy jest on kompletny (ilustracja 2.2). Następnie zapoznaj się z filmami instruktażowymi na stronie [1].

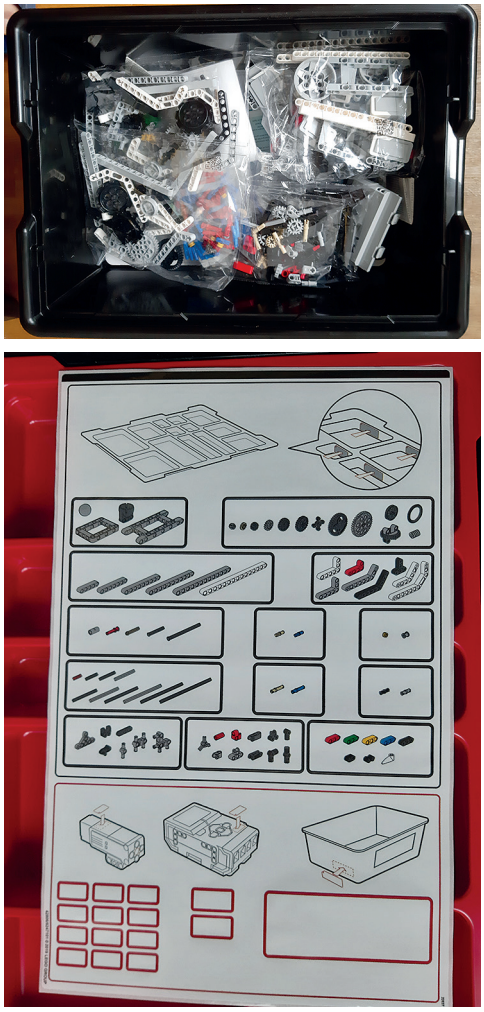
Rozpocznij od przyklejenia etykiet oraz rozpakowania klocków (ilustracja 2.3). Podczas rozpakowania powinieneś zapoznać się z kształtem klocków, aby później łatwiej zlokalizować potrzebny element. Trzymając wybrany klocek, spróbuj odpowiedzieć na pytanie, czy masz pomysł, do czego on może służyć?



Ilustracja 2.1. Pudełko z zestawem Lego Mindstorms EV3

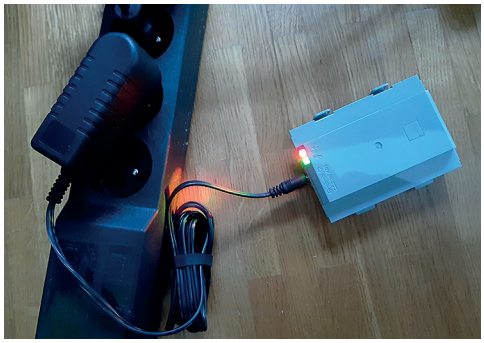
Jeżeli nie, nie przejmuj się, już wkrótce poznasz jego zastosowanie. Podobnie powinieneś postąpić z zestawem dodatkowych klocków.

W międzyczasie podłącz do ładowania baterię, jak to przedstawiono na ilustracji 2.4. Gdy dioda czerwona przestanie świecić, bateria będzie w pełni naładowana.



Ilustracja 2.2. Pudełko z zestawem Lego Mindstorms EV3 (wnętrze)

Ilustracja 2.3. Rozpakowany zestaw z poukładanymi klockami



Ilustracja 2.4. Ładowanie baterii. Gdy dioda czerwona przestanie świecić, bateria jest w pełni naładowana

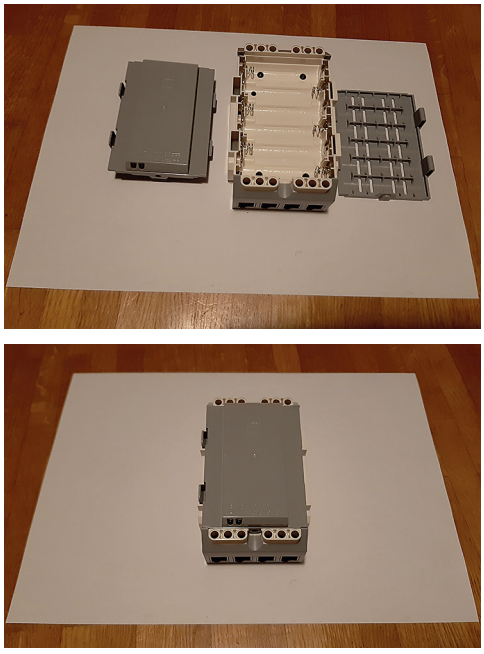
Następnie zlokalizuj programowalny klocek (inteligentna kostka) EV3, który będziemy nazywać kontrolerem (ilustracja 2.5). Klocek ten zawiera układ programowalny z procesorem ARM oraz niewielki system operacyjny umożliwiający tworzenie programów oraz konfigurację za pomocą przycisków na panelu oraz wyświetlacza monochromatycznego. Jest on swoistym komputerem, który komunikuje się z czujnikami i silnikami przy pomocy kabli. Możemy podłączyć kontroler do komputera za pomocą złącza USB, wczytać program z karty SD lub połączyć się przez Bluetooth.

Ilustracja 2.5. Programowalny klocek (kontroler) oraz dodatkowa bateria

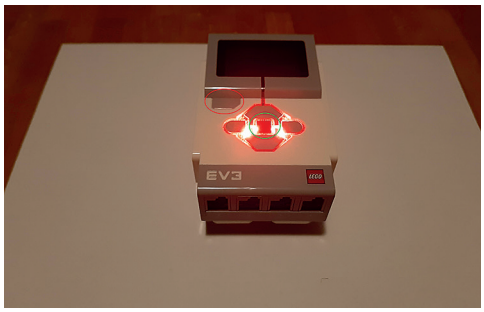
Obejrzyj dokładnie kontroler i spróbuj zlokalizować porty:

- oznaczone cyframi 1–4 służą do podłączania sensorów (Sensor). Dzięki nim kontroler odbiera informacje z otoczenia;
- oznaczane literami A–D służą do podłączenia serwomotorów (Motor). Dzięki nim kontroler wysyła informacje, uruchamiając różne silniki.

Na początku należy zamontować baterię do kontrolera, jak to pokazano na ilustracji 2.6. Następnie, w ramach testów, możesz spróbować włączyć zestaw, jak to pokazano na ilustracji 2.7. Gdy dioda zmieni kolor z czerwonego na zielony, zestaw jest gotowy do użycia. Używając przycisków na panelu, przejrzyj opcje na wyświetlaczu klocka, a następnie wyłącz kontroler.



Ilustracja 2.6. Podłączenie baterii



Ilustracja 2.7. Przycisk oznaczony zielonym kółkiem służy do włączenia kontrolera i akceptacji opcji (OK); przycisk oznaczony czerwonym kółkiem służy do wyjścia z wcześniej wybranej opcji w menu (Cancel), a otaczające go przyciski służą do poruszania się po menu

Znasz już wszystkie techniczne szczegóły, więc pora na zabawę i naukę. W kolejnych rozdziałach zdobędziesz wiedzę i umiejętności potrzebne do konstrukcji zaawansowanych robotów. Staraj się zwracać uwagę na wszystkie detale, a podczas ich nauki spróbuj wymyślić, w jakich sytuacjach można tę wiedzę zastosować – wiedza bez umiejętności jej zastosowania jest bezużyteczna.

Rozdział 3

Mechanika

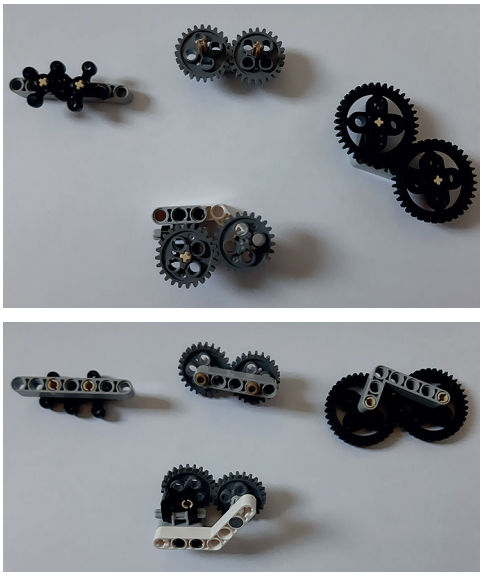
W tym rozdziale poznasz podstawowe zasady umożliwiające tworzenie ruchomych części oraz zamianę ruchu obrotowego silniczków na inne rodzaje ruchu. Jeżeli chcesz sobie przypomnieć pojęcia fizyczne, które będą w tym rozdziale stosowane, to zajrzyj do Dodatku B.

W poniższych sekcjach zajmiemy się konstrukcją prostych mechanizmów przy użyciu klocków.

3.1. Przekładnie

Przekładnie służą do przeniesienia napędu z elementu napędzającego do napędowego, często z modyfikacją jego parametrów. Elementy fizyki przekładni zostały omówione w Dodatku B. W tym rozdziale zajmiemy się ich praktycznym wykorzystaniem.

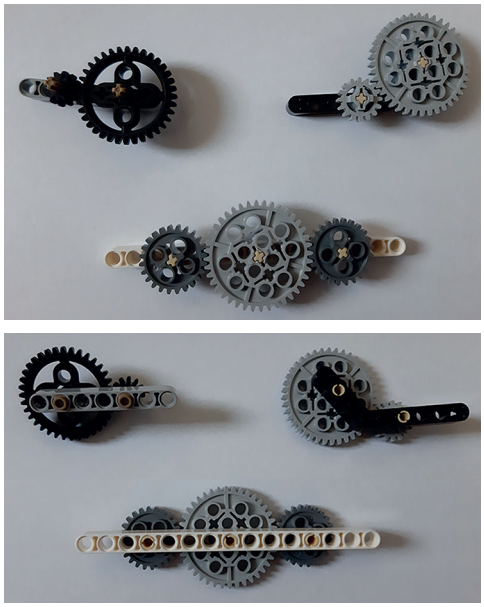
Spójrzmy teraz na przekładnie, które składają się z kół zębatych tych samych rozmiarów, zatem przełożenie jest w stosunku 1:1. Nie mamy więc zmiany szybkości obrotu i zysku momentu siły, jednak odwracamy kierunek obrotów i przenosimy ruch obrotowy w inne miejsce. Skonstruuj kilka takich przekładni – spróbuj zainspirować się ilustracją 3.1.



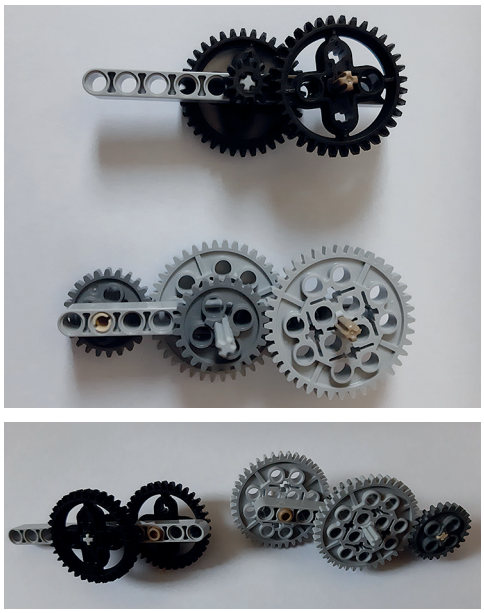
Ilustracja 3.1. Przekładnie 1:1

Kolejny przykład to przekładnie, w których używamy kół zębatych o różnej liczbie zębów. Spróbuj skonstruować przekładnie z ilustracji 3.2. Policz liczbę zębów każdego koła zębatego w przekładni, a następnie oblicz przełożenie takiego mechanizmu.

Koła zębate możemy również montować po obu stronach wału. Spróbuj wykonać przekładnie z ilustracji 3.3. Spróbuj obliczyć ich przełożenie. W którą stronę obracają się koła zębate?

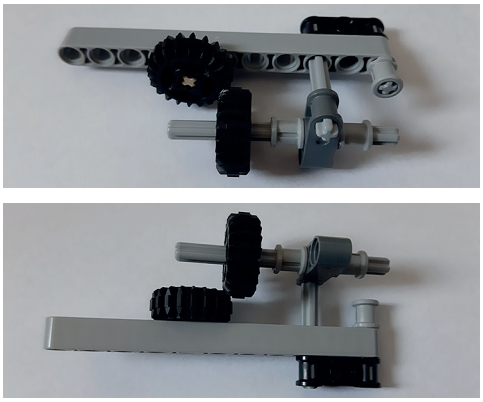


Ilustracja 3.2. Przykłady przekładni z kołami o nierównej liczbie zębów

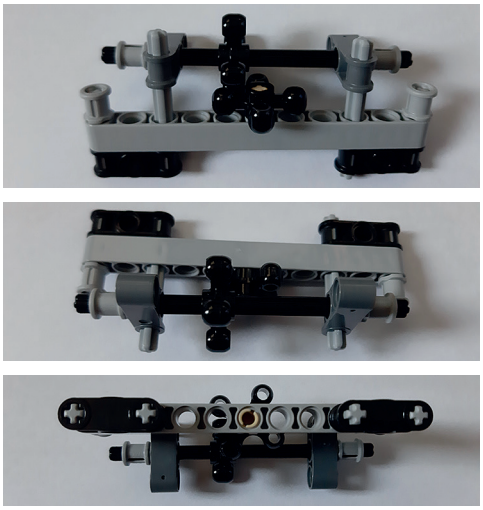


Ilustracja 3.3. Przykłady przekładni z kołami o nierównej liczbie zębów umieszczonymi po dwóch stronach belki

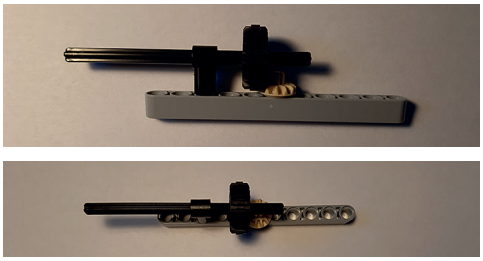
W kolejnym kroku spróbujmy przeanalizować przekładnie kątowe – takie, w których koła zębate są umieszczone pod pewnym kątem względem siebie. Przeanalizuj ilustracje 3.4, 3.5, 3.6, 3.7, a następnie skonstruuj odpowiednik tych przykładów. Pomyśl również, w jakiej sytuacji mógłbyś zastosować te przekładnie.



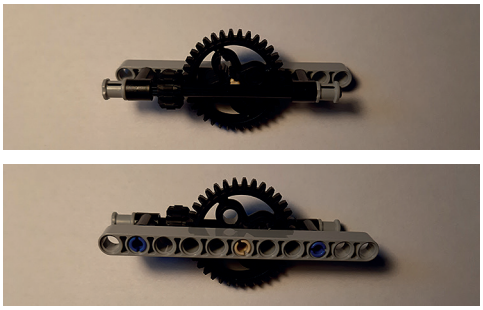
Ilustracja 3.4. Przykłady przekładni kątowej dla kąta 90°



Ilustracja 3.5. Kolejny przykład przekładni kątowej dla kąta 90°



Ilustracja 3.6. Następny przykład przekładni kątowej dla kąta 90°

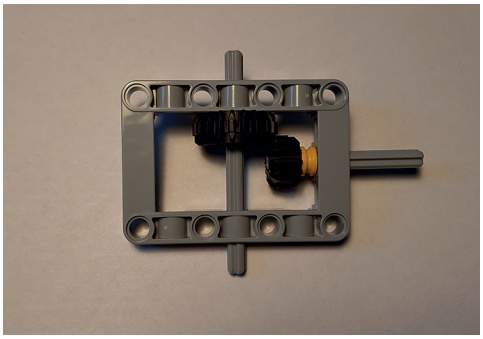


Ilustracja 3.7. Przykłady przekładni kątowej

3.1.1. Mechanizm różnicowy

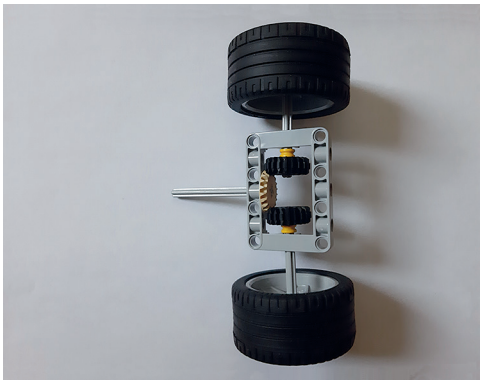
Teraz zajmiemy się pewnym szczególnym mechanizmem, który umożliwia w samochodzie przełożenie obrotów z silnika na koła w taki sposób, żeby cały mechanizm bez oporów mógł również skręcać. Mechanizm ten nazywa się mechanizmem różnicowym, a często również nazywany jest dyferencjałem.

Rozpocznijmy od bardzo prostej konstrukcji przełożenia obrotów przedstawionej na ilustracji 3.8. Jeżeli zamontujesz koła, to wówczas, napędzając taki mechanizm, będziesz mógł się poruszać do przodu, jeśli jednocześnie mechanizm zablokuje możliwość skrętu – delikatnie sprawdź, co się dzieje podczas skrętu kół.

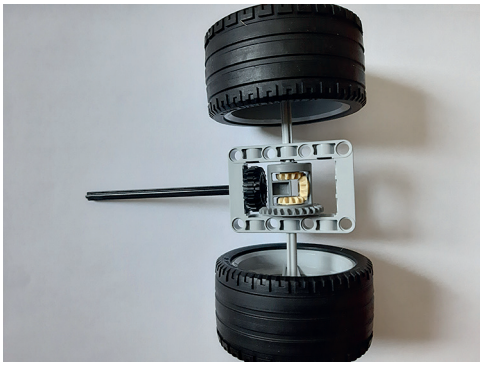


Ilustracja 3.8. Ruch do przodu – nie zapomnij zamontować kół

Zęby poprawić możliwość skrętu, skonstruuj mechanizm przedstawiony na ilustracji 3.9. Pozwala on napędzać koła, które wykonują skręt. Jednak problemem jest ruch do przodu. Czy potrafiłbyś połączyć te dwa poprzednie mechanizmy, aby układ miał możliwość ruchu do przodu, ale nie miał utrudnionego skrętu? Spróbuj taki mechanizm naszkicować na kartce, a następnie spróbuj go skonstruować. Taki mechanizm będzie wymagał użycia części z zestawu rozszerzającego. Spróbuj wykonać mechanizm z ilustracji 3.10.



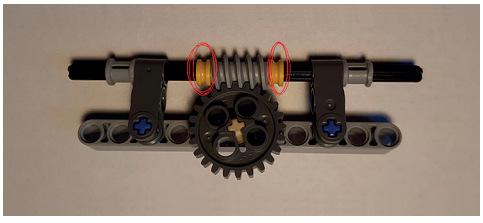
Ilustracja 3.9. Mechanizm umożliwiający skręt, ale nie ruch do przodu



Ilustracja 3.10. Mechanizm różnicowy umożliwiający ruch do przodu i skręt. Wymaga użycia klocków z zestawu dodatkowego

3.2. Śruba

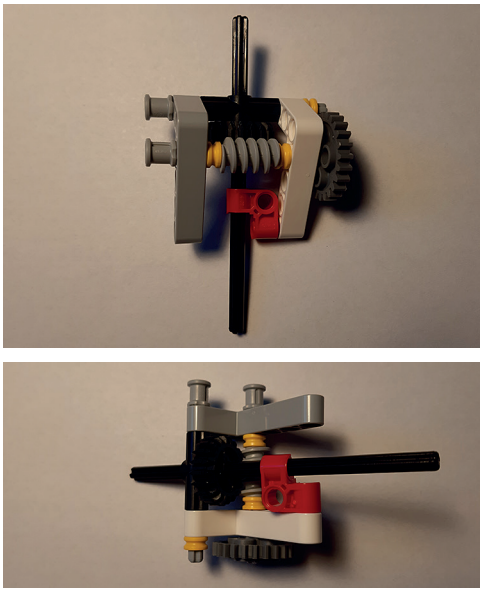
Śruba służy do zamiany ruchu obrotowego na ruch posuwisty postępowy, jak to opisano w Dodatku B. Jednak teraz podamy przykład przekładni ślimakowej, która umożliwia przełożenie ruchu obrotowego na obrotowy przy pomocy śruby. Mechanizm ten jest przedstawiony na ilustracji 3.11. Nie zapomnij o blokadach zaznaczonych czerwonymi elipsami. Spróbuj obrócić koło zębate i sprawdź, jak szybko obraca się wał, na którym jest śruba. Czy domyślasz się, dlaczego przekładnia ta nazywa się przekładnią ślimakową?



Ilustracja 3.11. Przekładnia ślimakowa z zaznaczonymi blokadami

Gdybyś usunął żółte blokady śruby, zaznaczone czerwonymi elipsami na ilustracji 3.11, otrzymałbyś rzeczywiste działanie śruby – zamianę ruchu obrotowego przekładni na ruch posuwisty.

Wariacją mechanizmu ślimakowego jest mechanizm przedstawiony na ilustracji 3.12.



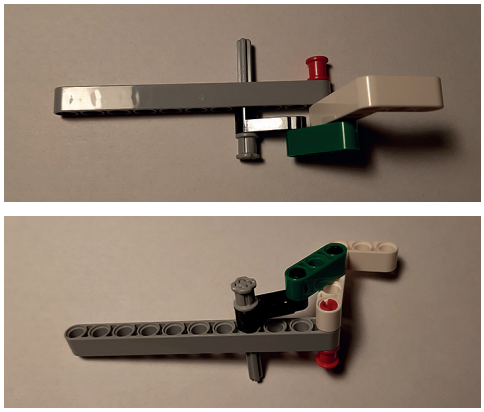
Ilustracja 3.12. Inny przykład przekładni ślimakowej

3.3. Zamiana ruchu obrotowego na inne formy

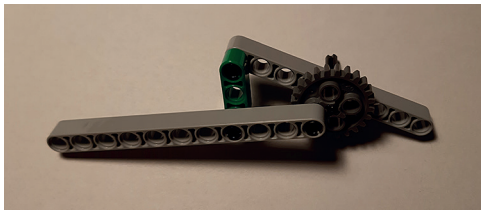
Silniki dostarczone do zestawu generują ruch obrotowy. Jednak często chcielibyśmy zamienić go na inne formy ruchu. W tej części przedstawimy przykłady mechanizmów umożliwiających taką transformację.

3.3.1. Mimośród

Rozpoczniemy od budowy mechanizmu przedstawionego na ilustracji 3.13. Próbuj obrócić wał (czarny podłużny klocek). Czy widzisz ograniczenia tego mechanizmu? Spróbujemy go ulepszyć, używając konstrukcji nazywanej mimośrodem. Tym pojęciem określa się obracające się koło (np. tryb), do którego przyczepiona jest, poza środkiem koła, część mogąca swobodnie się obracać. Układ taki stanowi korbę. Przykład takiego mechanizmu przedstawiono na ilustracji 3.14.

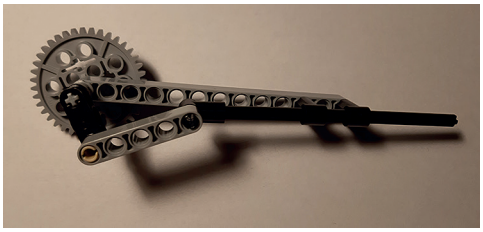


Ilustracja 3.13. Przykład mechanizmów do zamiany ruchu obrotowego na postępowy

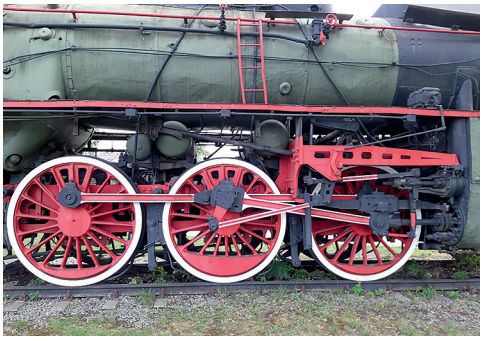


Ilustracja 3.14. Mimośród

Przejdźmy teraz do innego sposobu zamiany ruchu obrotowego na postępowy. Kolejny mechanizm przedstawiono na ilustracji 3.15. Jeżeli miałeś okazję widzieć lokomotywę parową, to z pewnością widziałeś go na żywo, gdyż jest on używany do napędzania kół lokomotywy (ilustracja 3.16).



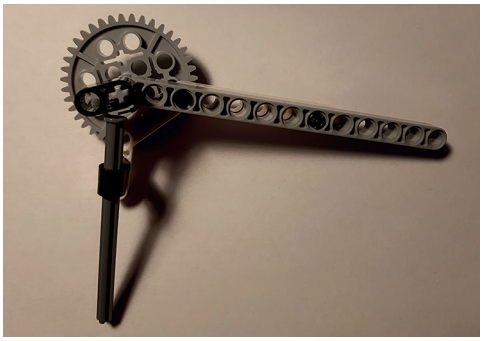
Ilustracja 3.15. Mimośród, inny przykład



Ilustracja 3.16. Zastosowanie mimośrodu do napędu lokomotywy parowej na przykładzie zabytkowej lokomotywy w Przeworsku

3.4. Krzywki

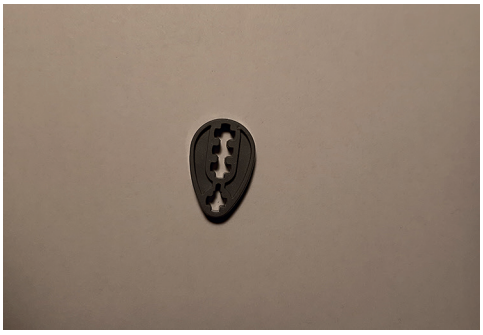
Inny sposób zamiany ruchu obrotowego na postępowy przedstawia mechanizm z częścią zwaną krzywką. Jest to wydłużony element, który zamontowany na wale obrotowym popycha rytmicznie inny wał. Przykład konstrukcji mechanizmu z krzywką przy użyciu klocków z zestawu podstawowego przedstawiony jest na ilustracji 3.17.



Ilustracja 3.17. Przykład konstrukcji przy użyciu krzywki (czarny podłużny klocek popychający pałeczkę w czarnej obejmie)

Skonstruuj ten mechanizm i przeanalizuj jego działanie. Czy taki mechanizm mógłby działać w środowisku, w którym siła grawitacji nie występuje?

W zestawie rozszerzonym znajduje się znacznie lepsza krzywka, która została przedstawiona na ilustracji 3.18. Spróbuj nią zastąpić krzywkę (czarny klocek) z poprzedniego rysunku.



Ilustracja 3.18. Krzywka z zestawu dodatkowego

3.5. Podsumowanie

Poznałeś już wystarczająco dużo informacji z teorii budowy mechanizmów i zasad ich działania, aby móc z nich samodzielnie skonstruować bardziej złożone układy. Poświęć teraz chwilę, aby powtórzyć to, co już umiesz, a następnie spróbuj stworzyć nowy mechanizm, który będzie użyteczny.

Rozdział 4

Programowanie

W tym rozdziale poznasz podstawy programowania, które przydadzą się w kolejnych rozdziałach, gdy będziesz już konstruował roboty. Szczegóły instalacji oprogramowania znajdziesz w załączniku C. Skupimy się teraz na wyjaśnieniu, czym jest programowanie i jak programować zestaw Lego.

4.1. Teoria

Programowanie polega na stworzeniu przepisu na wykonanie konkretnego zadania. Ten przepis nazywa się algorytmem i jest on zapisywany w języku programowania rozumianym przez komputer lub urządzenie. Algorytm w takiej formie nazywa się programem.

Jednym z najprostszych algorytmów zapisanych w języku zrozumiałym dla ludzi jest przepis na ciasto. Zawiera on listę składników i sekwencję kroków, które należy wykonać, żeby ciasto upiec. Przepis ten jest bardzo ogólny ze względu na to, że człowiek jest w stanie rozumować na takim poziomie. Dla przykładu, stwierdzenie o pieczeniu przez, powiedzmy, 30 minut nie zawiera informacji o tym, ze należy otworzyć piekarnik, włożyć ciasto, zamknąć piekarnik, a także informacji o dokładnym ustawieniu konkretnego modelu piekarnika. Dla człowieka, który ma odpowiednie doświadczenie kulinarne, są to oczywiste rzeczy.

Inaczej się rzecz ma w przypadku programowania komputera lub ogólnie programowalnego urządzenia elektronicznego. Tutaj twórcy przepisu muszą wykazać się dużą skrupulatnością, aby dokładnie podać wszystkie informacje, z których komputer złoży pełny i dokładny schemat działania. Pamiętaj, że komputer nie potrafi się domyślić intencji programisty, więc jeżeli pominiesz istotny szczegół lub po prostu o nim zapomnisz, komputer z pewnością zgłosi błąd.

Absolutnie nie powinieneś się przejmować tym, że popełniłeś błąd. Podczas programowania napisanie programu zajmuje bardzo mało czasu w porównaniu do późniejszego znalezienia wszystkich błędów. W przypadku złożonych programów praktycznie nigdy nie zdarza się tak, aby program działał za pierwszym razem. Programista stosuje metodyczne podejście do szukania błędów. Istnieje zawód tester oprogramowania, który skupia się na dokładnym sprawdzeniu,

czy program działa tak, jak powinien. Dlatego staraj się podczas pisania programów wyrabiać sobie nawyk analizy tego, co stworzyłeś, szukania błędów i ich szybkiej naprawy.

Istnieje wiele podejść do programowania – informatycy nazywają je paradygmatami programowania. Mamy więc:

- paradygmat imperatywny – program jest napisany jak lista zadań, gdzie programista krok po kroku dokładnie opisuje, co komputer powinien zrobić;
- paradygmat deklaratywny – programista określa cel, jaki powinien osiągnąć komputer, a niekoniecznie przepis, jak ten cel osiągnąć – to zadanie dla komputera; programowanie w paradygmacie deklaratywnym wymaga interpretera, który taki cel potrafi zrealizować na podstawie deklaracji programisty.

Inny podział można wyróżnić ze względu na konstrukcje używane w programowaniu:

- paradygmat proceduralny – program jest opisem krok po kroku poleceń, które składają się na dany algorytm; często powtarzające się fragmenty programu grupuje się w procedurach/funkcjach;
- paradygmat obiektowy – dane (atrybuty) i operacje (metody) na nich wykonywane są zorganizowane w „strukturach" zwanych klasami; obiekty są klasami wypełnionymi konkretnymi danymi – nazywane są instancjami klasy;
- paradygmat funkcyjny – opiera się na funkcjach rozumianych w sensie matematycznym – funkcja zawsze zwraca tę samą wartość dla ustalonego jej argumentu[1]; języki funkcyjne służą często do rozwiązywania skomplikowanych problemów programistycznych w abstrakcyjny i bliski matematyce sposób.

Wiele języków programowania służy do programowania w kilku paradygmatach jednocześnie.

Wspomnieć trzeba jeszcze o podziale ze względu na sposób uruchamiania. Wyróżniamy języki:

- interpretowane – program w języku jest interpretowany, jeżeli wykonuje go specjalny program zwany interpreterem;
- kompilowane:
  - do kodu maszynowego – program jest przetwarzany na sekwencję instrukcji dla procesora.
  - do kodu pośredniego (tzw. *bytecode*) – program jest kompilowany do kodu pośredniego, który wykonuje specjalny interpreter.

[1] W przypadku paradygmatu proceduralnego, w którym wykorzystywane są również „funkcje", nie zawsze jest to prawdą. Jeżeli wykorzystujemy „niematematyczną funkcję", np. taką, która odczytuje klawisz naciśnięty przez użytkownika, to każde jej wywołanie z tymi samymi argumentami może zwrócić inną wartość (naciśnięty klawisz). Czy to oznacza, że w językach funkcyjnych nie możemy wchodzić w interakcje z otoczeniem? Nie, opracowano matematykę mechanizmów obsługi takich „niematematycznych funkcji". Opiera się ona na teorii kategorii. Jeżeli jesteś zainteresowany, to spróbuj dowiedzieć się więcej o terminie „Monada" w programowaniu funkcyjnym.

Programy napisane w językach kompilowanych do kodu maszynowego działają szybko, gdyż wynikowy program jest bezpośrednio wykonywany na sprzęcie. Natomiast języki interpretowane są wolniejsze, gdyż występuje narzut związany z tłumaczeniem instrukcji języka na instrukcje procesora przez interpreter.

Jednak zaletą języków interpretowanych jest to, że interpreter sprawia, że język taki może być bardziej wysokopoziomowy.

Kiedyś język programowania był zależny od architektury sprzętu. Takie języki nazwano językami niskiego poziomu. Program napisany dla określonego typu komputera nie mógł zostać łatwo przeniesiony na inny typ komputera. Ten stan trwał do lat 70., gdy powstały dwa języki wysokiego poziomu, które uniezależniały proces programowania od specyficznej budowy komputerów. Pierwszym takim językiem był FORTRAN (ang. *FORmula TRANslator*). Był to język przeznaczony głównie do obliczeń naukowy. Współcześnie nadal używa się jego nowszych wersji. Drugim językiem wysokiego poziomu był LISP (ang. *LISt Processor*), który opierał się głównie na paradygmacie funkcyjnym, chociaż umożliwiał też programowanie proceduralne, a później również zaawansowane programowanie obiektowe. LISP miał ciekawą cechę – program w tym języku był listą – strukturą danych, do manipulacji którą ten język stworzono. Zatem bez większego wysiłku można było napisać samomodyfikujący się program. Z tego języka równie często korzysta się dzisiaj, szczególnie w zaawansowanych technologiach takich jak sztuczna inteligencja.

Następnie w wyniku eksperymentów z różnymi podejściami do programowania powstała cała gama języków. Istotnym zwrotem w historii języków programowania, było użycie języka C do napisania systemu operacyjnego Unix. System ten został użyty do obsługi węzłów (serwerów) nowej sieci ARPANET łączącej superkomputery w dużych ośrodkach akademickich i wojskowych w USA, która później stała się siecią Internet. To przyczyniło się do dużej popularności języka C, a następnie jego rozszerzenia C++.

Obecnie istnieje wiele języków programowania. Poniżej wymieniono tylko kilka popularniejszych:

- C/C++ – imperatywny język kompilowany do kodu maszynowego, używany w programowaniu na poziomie systemu operacyjnego lub szybkich obliczeń. Język C++ był początkowo obiektowym rozszerzeniem języka C. Ten drugi umożliwiał jedynie programowanie proceduralne;
- Java – imperatywny język kompilowany do bytecode'u, bardzo podobny do C++. Opiera się na paradygmacie obiektowym. Jest on niezależny od architektury ze względu na użycie maszyny wirtualnej – interpretera, który wykonuje program w Javie;
- Python – imperatywny język interpretowany, stworzony do nauki programowania, ale obecnie wyko-

rzystywany w każdym celu – prototypowania koncepcji, automatyzacji zadań administracyjnych, obliczeń naukowych, analizy danych i uczenia maszynowego oraz wielu innych. Umożliwia programowanie proceduralne, obiektowe oraz funkcyjne. Posiada ogromną liczbę różnorodnych bibliotek, przez co stał się bardzo popularny;

- Haskell – jest to czysto funkcyjny język imperatywny, interpretowany, ale również kompilowany. Nie jest aż tak popularny jak powyższe języki, ale jest dobrym punktem startowym do nauki programowania funkcyjnego i teoretycznego podejścia do informatyki;
- SQL – (ang. *Structured Query Language*), Strukturalny Język Zapytań dla (relacyjnych) baz danych. Jest to język deklaratywny wykorzystywany do interakcji z bazą danych. Umożliwia konstrukcję zapytań o dane kierowanych do bazy danych. Zadaniem bazy danych jest wydobycie danych spełniających kryteria zapisane przez programistę w tym języku. Jest on niezwykle popularny ze względu na to, że bazy danych obecnie stanowią podstawę systemów sieciowych, np. sklepów internetowych, systemów bankowych i rządowych.

Lista języków programowania jest bardzo długa. Należy jednak pamiętać, że to język należy dobierać pod kątem zastosowań, a nie odwrotnie. Język jest jedynie narzędziem, a nie celem samym w sobie. Dobrze dobrany potrafi znacznie ułatwić zadanie programistyczne.

Po tym krótkim omówieniu idei programowania i historii języków programowania przejdziemy do naszego głównego celu – programowania klocków LEGO® Mindstorms® EV3.

4.2. Przygotowanie

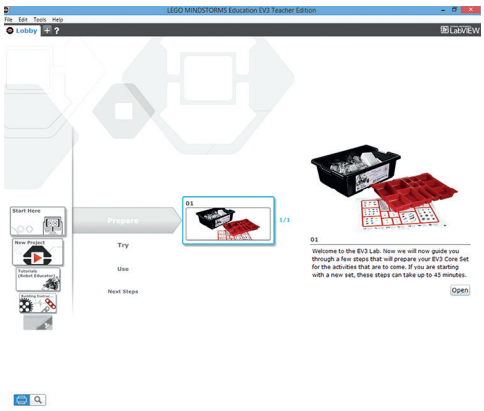
Początek przygody z programowaniem rozpocznij od uruchomienia programu LEGO Mindstorms EV3 Software. Szczegóły instalacji na różnych systemach operacyjnych i w różnych konfiguracjach znajdziesz w Dodatku C.

Kliknięcie odpowiedniej ikony (ilustracja 4.1) uruchomi program. Po uruchomieniu okno programu wygląda domyślnie tak jak na ilustracji 4.2. Poświęć chwilę, aby przejrzeć wszystkie możliwe opcje w części startowej.

Ilustracja 4.1. Ikona programu do programowania

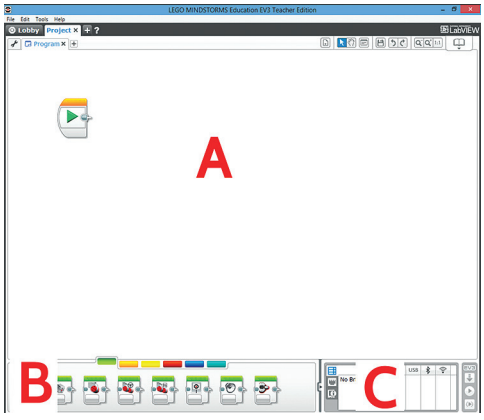
Czas rozpocząć programowanie. W tym celu wybierz File → New Project → Program. Wówczas zobaczysz okno jak na ilustracji 4.3. Programowanie polega

na przeciąganiu elementów z obszaru B na obszar A i układaniu z nich powiązanych logicznie instrukcji. Spróbuj przeglądnąć wszystkie kolorowe zakładki w obszarze B, aby zaznajomić się z instrukcjami, których możesz użyć.



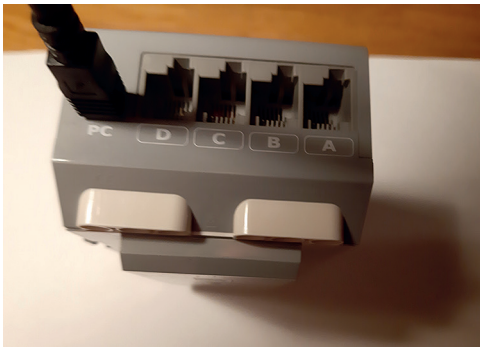
Ilustracja 4.2. Początkowe okno programu

Kontroler podłącz do komputera przy pomocy kabla USB z zestawu (porównaj z ilustracją 4.4). Drugi koniec podłącz do wolnego portu USB w komputerze.

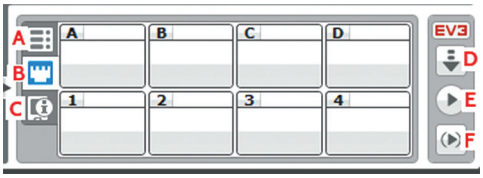


Ilustracja 4.3. Nowy program. A – obszar, w którym tworzymy program; B – zasobnik instrukcji języka programowania; C – informacje o podłączonym kontrolerze

Klocek powinien być włączony. Jeżeli program wykryje klocek, to część okna C z ilustracji 4.3 zmieni się na to z ilustracji 4.5.



Ilustracja 4.4. Podłączenie kabla USB do kontrolera



Ilustracja 4.5. Wykrycie klocka w programie: A – informacje o klocku; B – widok podłączenia urządzeń do portów klocka; C – dostępne klocki; D – przesłanie programu do klocka; E – przesłanie programu i uruchomienie go na klocku; F – uruchomienie wybranego programu

Wygodniejsze jest podłączenie klocka do komputera przy użyciu Bluetootha, które zostało dokładnie opisane w Dodatku C.

Ze względu na to, że budujesz ruchome mechanizmy, staraj się przesyłać programy do klocka D, a następnie po odłączeniu kabla od komputera możesz taki program uruchomić na klocku.

4.3. Opis języka programowania

Gdy znasz już teorię, spróbuj zaznajomić się z graficznym językiem programowania klocków.

Poniżej opisano skrótowo bloki znajdujące się na każdej z zakładek. Możesz w każdej chwili otrzymać podpowiedź o tym, do czego służy dany element, najeżdżając na niego myszką. Pojawi się wówczas krótka informacja o funkcji elementu.

4.3.1. Zielony

Na zielonej zakładce (ilustracja 4.6) znajdziesz instrukcje związane z programowaniem elementów aktywnych – silników, wyświetlacza oraz głośnika wbudowanego w klocek. Bloki z tej zakładki będą opisane dokładniej w kolejnym rozdziale, gdy będziesz konstruował mechanizmy aktywne.

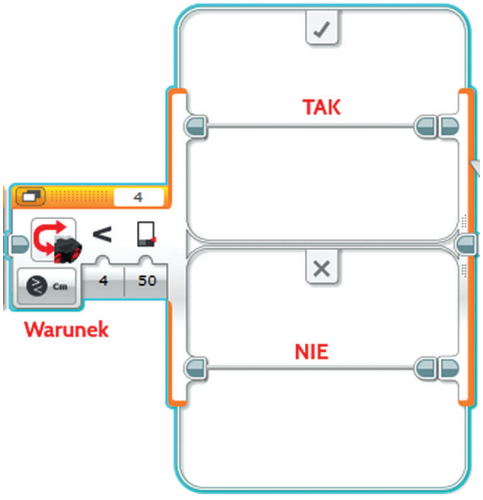
Ilustracja 4.6. Panel zielony

4.3.2. Pomarańczowy

Na zakładce pomarańczowej (ilustracja 4.7) znajdziesz instrukcje pomagające kontrolować przepływ sterowania w programie – podejmiesz decyzję lub zrobisz pętlę.

Ilustracja 4.7. Panel pomarańczowy

Skupimy się tutaj na dwóch użytecznych blokach. Pierwszy z nich to instrukcja podejmowania decyzji – SWITCH (IF) przedstawiona na ilustracji 4.8.

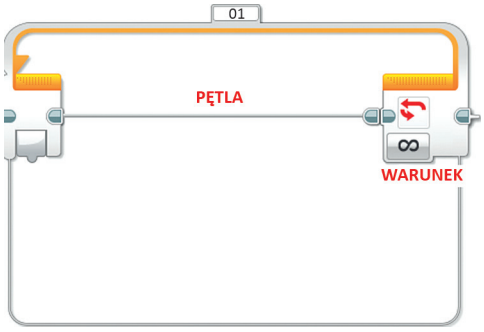


Ilustracja 4.8. Instrukcja służąca do podejmowania decyzji. Jeżeli warunek jest spełniony, zostanie wykonany ciąg instrukcji umieszczony na linii TAK, w przeciwnym wypadku ciąg instrukcji dla NIE

Drugą ważną instrukcją jest instrukcja pętli – LOOP – przedstawiona na ilustracji 4.9. Szczególnie istotne jest powtórzenie pętli nieskończoną liczbę razy. Służy to do zapewnienia działania programu w nieskończoność.

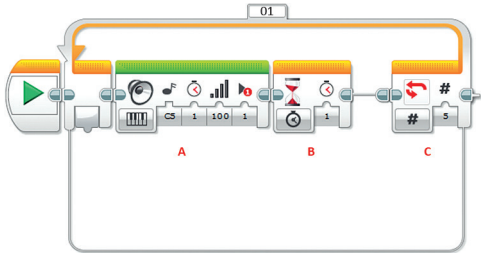
Jako prosty przykład możemy podać program, który odtwarza wybrany ton. Opis programu (tzw. pseudokod) jest następujący:

- Pętla powtórzona 5 razy:
  - odtwórz wybrany dźwięk,
  - odczekaj 1 sekundę.



Ilustracja 4.9. Instrukcja służąca do iteracji – powtarzania instrukcji znajdujących się na linii pętli. Będą one powtarzane, jeżeli warunek będzie spełniony. W przypadku wyboru symbolu nieskończoności ∞ pętla będzie działała w nieskończoność

Spróbuj to samemu zaimplementować, przenosząc odpowiednie bloki na obszar roboczy. Jeżeli nie wiesz, jak to zrobić, zobacz program z ilustracji 4.10.



Ilustracja 4.10. Program w pętli C, wykonującej się 5 razy, robi następujące rzeczy: A – gra ton C5; B – czeka jedną sekundę

4.3.3. Żółty

Na zakładce żółtej (ilustracja 4.11) znajdziesz instrukcje związane z odczytami z czujników.

Ilustracja 4.11. Panel żółty

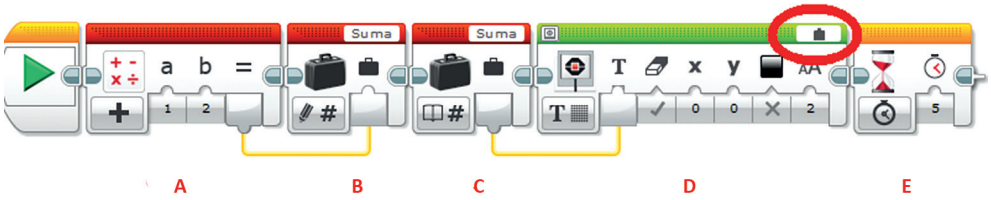
4.3.4. Czerwony

Na zakładce czerwonej (ilustracja 4.12) znajdziesz instrukcje związane z manipulacją danymi. Spróbuj napisać program obliczający sumę dwóch liczb, a następnie wyświetlający ją na ekranie kontrolera. Program przedstawiony jest na ilustracji 4.13. Po jego uruchomieniu powinieneś mieć możliwość odczytania liczby 3 na ekranie kontrolera.

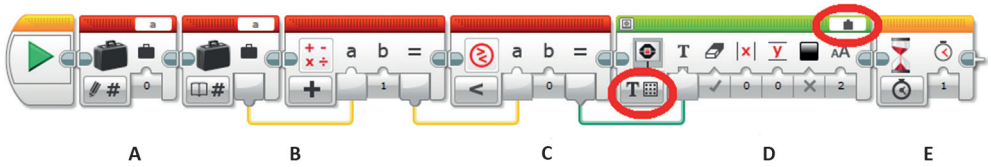
Żeby oswoić się ze zmiennymi, napisz program według wzoru przedstawionego na ilustracji 4.14. Prezentuje on przykład użycia zmiennej do porównywania wartości zmiennej.

Ilustracja 4.12. Panel czerwony

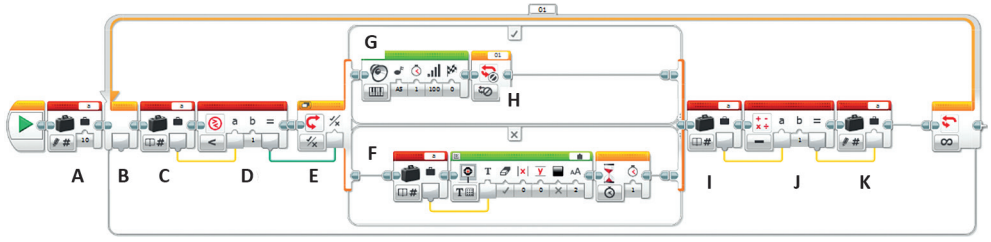
Na koniec przygody ze zmiennymi, pętlami i przepływem sterowania stwórz bardziej zaawansowaną pętlę, taką jak ta przedstawiona na ilustracji 4.15. Program wypisuje na ekranie kontrolera kolejne liczby od 10 do 1, a następnie odtwarza dźwięk i kończy działanie. Spróbuj go skonstruować zgodnie z rysunkiem i uruchomić. Ze względu na zawiłość tego kodu rozpisano go dokładniej, a ciebie zachęcam do dokładnego jego przeanalizowania, gdyż znajdziesz w nim wiele ciekawych rozwiązań.



Ilustracja 3.13. Program sumujący liczby i wyświetlający wynik na ekranie. A – zsumuj 1 + 2; B – odczytaj zmienne numeryczne zmiennej „Suma"; C – wypisz zmienną „Suma"; D – aby połączenie poprzedniego bloku było możliwe, kliknij na miejsce oznaczone czerwonym kółkiem i wybierz *Wired*; E – zaczekaj 5 sekund zanim zakończysz program, abyś mógł odczytać zawartość ekranu



Ilustracja 4.14. A – stwórz zmienną *a* typu numerycznego i podstaw do niej wartość 0; B – odczytaj wartość zmiennej *a* (również 0) i dodaj do niej 1; C – sprawdź, czy wynik dodawania ($a + 1 = 1$, dla $a = 0$) jest mniejszy od 0. Wartość logiczną takiego porównania (prawdę interpretujemy jako 1, a fałsz jako 0) wypisz w ekran D. Pamiętaj, aby w górnym rogu zielnego bloku zaznaczyć *Wired* (zobacz czerwoną obwódkę), gdyż tylko wtedy możesz podłączyć poprzedni czerwony blok; E – opóźnienie czasowe, które umożliwi Ci odczytanie wartości logicznej na ekranie klocka, zanim program się zakończy



Ilustracja 4.15. A – utwórz zmienną numeryczną o nazwie *a* i podstaw do niej wartość 10; B – otwórz nieskończoną pętlę, którą w praktyce przerwiemy, gdy zmienna *a* osiągnie wartość 0, o czym poniżej; C – pobierz aktualną wartość zmiennej *a*, a następnie sprawdź, czy ma ona wartość mniejszą od 1 – D; następnie w instrukcji E podejmij na podstawie wartości logicznej tego porównania; F – jeżeli aktualna wartość zmiennej *a* jest większa od 1, to wypisz ją na ekranie podobnie jak w poprzednim przykładzie; G – jeżeli natomiast $a < 1$, to odtwórz dźwięk, następnie przerwij nieskończoną pętlę (B) przy pomocy instrukcji reprezentowanej przez blok H, co kończy program, gdyż za nieskończoną pętlą nie ma żadnego innego bloku. Za introdukcją E kryje się ciąg bloków, który służy do pobierania aktualnej wartości zmiennej *a* (blok I), zmniejsz wartość zmiennej *a* o 1 w bloku J, a na końcu tę nową wartość ponownie zapisz do zmiennej *a* w bloku K

- $a \leftarrow 10$ – utwórz numeryczną zmienną a i podstaw do niej wartość 10.
- Nieskończona pętla:
  - Sprawdź, czy $a < 1$:
    - TAK: Zagraj dźwięk i opuść nieskończoną pętlę.
    - NIE: Wypisz wartość zmiennej a na ekranie i odczekaj chwilę.
  - $a \leftarrow a + 1$ – pobierz wartość zmiennej a, dodaj do niej 1, a następnie wynik podstaw ponownie do zmiennej a.

4.3.5. Niebieski

Na zakładce niebieskiej (ilustracja 4.16) znajdziesz instrukcje zaawansowane. Na początkowym etapie nauki nie będziesz potrzebował elementów z tej części.

Ilustracja 4.16. Panel niebieski

4.4. Podsumowanie

Poznałeś ideę konstrukcji programów do klocków LEGO® Mindstorms® EV3. W kolejnym rozdziale poznasz metody pisania programów, które wykorzystują odczyty z czujników i aktywują silniki.

Klocek możesz również przystosować do programowania w języku Python. Wymaga to nagrania na kartę pamięci SD dodatkowego systemu operacyjnego i uruchomienia go na klocku. Wówczas możesz programować przy użyciu programu Microsoft Visual Studio ze specjalną wtyczką programistyczną. Procedura opisująca konfigurację znajduje się na stronie internetowej LEGO [3]. Jest to podejście dające znacznie więcej możliwości pod warunkiem, że nauczysz się programować w języku Python. Jest to język programowania często wykorzystywany obecnie w różnych dziedzinach informatyki, a więc na pewnym etapie warto poświęcić chwilę i się go nauczyć. Możesz znaleźć wiele książek łagodnie wprowadzających do pisania programów w Pythonie, np. [4].

Warto również wspomnieć, że możemy uruchomić pełny system Linux na kontrolerze, zapisując go na karcie SD i uruchamiając z niej klocek. Procedura jest opisana na stronie internetowej [5]. Daje to pełną kontrolę nad klockiem i możliwość programowania w dowolnym języku programowania. Jest to podejście niestandardowe, które wymaga pewnej zaawansowanej wiedzy informatycznej. Rozwiązanie to nie jest również oficjalne, przez to nie zalecamy jego stosowania.

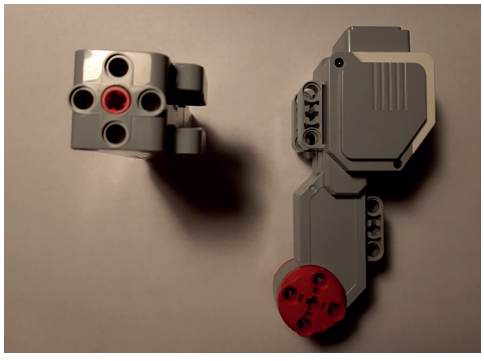
Rozdział 5

Czujniki i silniki

W tym rozdziale poznasz podstawy programowania silników i czujników, aby dać robotowi, którego skonstruujesz w kolejnym rozdziale, możliwość poruszania się i poczucia otaczającego go świata.

5.1. Opis silników

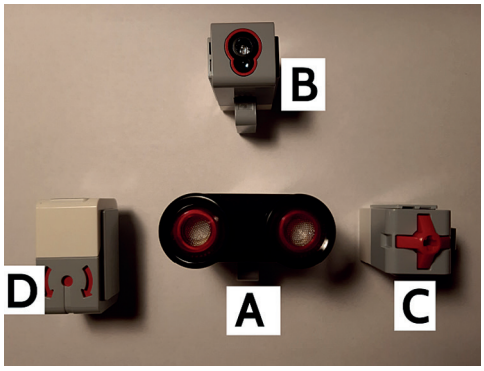
W zestawie masz do dyspozycji dwa silniki o większej mocy i jeden silnik precyzyjny (ilustracja 5.1). Silniki podłączamy do portów alfanumerycznych *A*–*D*. Programowanie ich omówimy poniżej, podczas omawiania programowania czujników.



Ilustracja 5.1. Zestaw silników. Prawy to silnik o większej mocy, a po lewej jest silnik precyzyjny

5.2. Opis czujników

Czujniki przedstawiono na ilustracji 5.2. Na początku omówimy fizyczne zasady działania takich czujników.



Ilustracja 5.2. Zestaw czujników: A – ultradźwiękowy czujnik odległości; B – czujnik kolorów; C – czujnik dotykowy; D – żyroskop

5.2.1. Ultradźwiękowy czujnik odległości

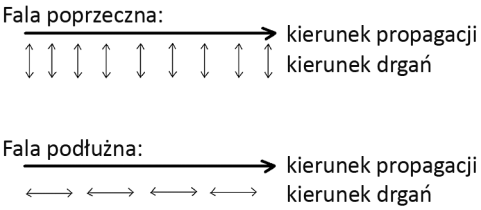
Fala dźwiękowa jest podłużną falą zmian ciśnienia propagującą się w powietrzu. Rozróżniamy fale:

- poprzeczne – drgania odbywają się prostopadle do kierunku poruszania się fali. Przykładem takich fal są fale w gumowym wężu poru-

szanym w górę i w dół. Tutaj nośnikiem jest gumowy wąż. Innym przykładem jest fala elektromagnetyczna, w której prostopadle do kierunku ruchu drga naprzemiennie pole elektryczne i magnetyczne, wzajemnie się indukując[2];

- podłużne – fala drga w kierunku ruchu. Przykładem jest wspomniana fala dźwiękowa w powietrzu lub w ciele stałym.

Przedstawiono je na ilustracji 5.3.



Ilustracja 5.3. Typy fal

Z falą kojarzymy jej prędkość propagacji – c – mierzoną w jednostkach prędkości, np. $\frac{\mathrm{m}}{\mathrm{s}}$; oraz częstotliwość – ν – liczbę drgań, którą zarejestruje czujnik na jednostkę czasu mierzoną w jednostkach odwrotności czasu, np. $[\nu] = \frac{1}{\mathrm{s}} = \mathrm{Hz}$, gdzie Hz to herc.

Ultradźwięki to fale o odpowiednio wysokiej częstotliwości, które są niesłyszalne dla człowieka. Każdy z nas ma tę górną granicę inną, ale umownie przyjmuje się, że ultradźwięki rozpoczynają się od częstotliwości 20 kHz. Niektóre zwierzęta, jak psy czy nietoperze, słyszą ultradźwięki. Co więcej, nietoperze i delfiny używają ich do orientacji w terenie i połowów. Nasz czujnik odległości działa na podobnej zasadzie.

Zastanówmy się teraz, jak możemy zmierzyć odległość przy użyciu ultradźwięków. Znamy prędkość (ultra) dźwięków w powietrzu[3] c i potrafimy mierzyć różnicę czasu pomiędzy wysłaniem impulsu i jego powrotem po odbiciu od przeszkody Δt. Wówczas drogę, którą przebiegnie taki sygnał, możemy określić z prostego wzoru

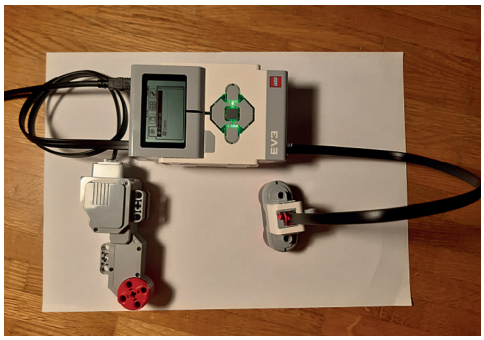
$$s = c\Delta t. \qquad (5.1)$$

Ponieważ odległość od czujnika do przeszkody sygnał przebiega dwa razy (drogę czujnik–przeszkoda–czujnik), odległość do przeszkody to $l = \frac{s}{2}$.

[2] Zmienne pole elektryczne indukuje (wirowe) pole magnetyczne, a zmienne pole magnetyczne indukuje (wirowe) pole elektryczne. Jest to zjawisko opisywane równaniami Maxwella w elektrodynamice. Zjawisko indukcji elektromagnetycznej jest kluczowe do zrozumienia wytwarzania prądu w różnego rodzaju prądnicach i generatorach.

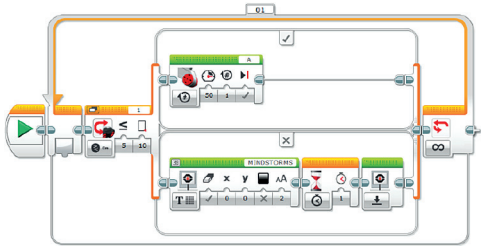
[3] Prędkość dźwięku w powietrzu zależy od parametrów termodynamicznych, szczególnie temperatury. Przy temperaturze 15°C wynosi ona $c \approx 340\ \frac{\mathrm{m}}{\mathrm{s}}$. Zatem po 3 sekundach dźwięk przebędzie około 1000 m = 1 km. Możesz wykorzystać tę zależność do przybliżonego pomiaru odległości wyładowań elektrycznych podczas burzy. Ponieważ prędkość światła jest bardzo duża w porównaniu z dźwiękiem, możesz, z (użytecznym) przybliżeniem, założyć, że światło rozchodzi się natychmiastowo. Jeżeli więc zobaczysz błysk, zacznij odliczać czas w sekundach do usłyszenia grzmotu. Po podzieleniu czasu przez 3 otrzymasz przybliżoną odległość w kilometrach do zaobserwowanego wyładowania.

Na początku spróbuj podłączyć czujnik ultradźwiękowy do kontrolera (później podłączymy motor), jak na ilustracji 5.4. Sprawdź na kontrolerze, czy czujnik mierzy odległość.



Ilustracja 5.4. Podłączenie czujnika ultradźwiękowego i motoru

Następnie podepnij motor do kontrolera, jak na ilustracji 5.4. Następnym etapem jest napisanie programu, który uruchamia silnik w przypadku wykrycia przeszkody. Jest on przedstawiony na ilustracji 5.5.



Ilustracja 5.5. Program sterujący czujnikiem ultradźwiękowym i motorem

Działanie jest następujące:

- Nieskończona pętla:
  - Czy odległość do przeszkody jest mniejsza niż 10 cm?
    - TAK: Uruchom silnik, wykonując jeden obrót.
    - NIE: Wypisz na ekranie „MINDSTORMS", następnie poczekaj 1 sekundę, a później wyczyść ekran.

Spróbuj poeksperymentować z czujnikiem, zanim przejdziesz do kolejnego kroku. Czy potrafiłbyś powiedzieć, w jakich sytuacjach taki czujnik mógłby mieć zastosowanie?

5.2.2. Czujnik kolorów

Spróbuj wykonać podobny eksperyment do opisanego w poprzednim rozdziale czujnika kolorów – zobacz ilustrację 5.2. Zastanów się, w jakich sytuacjach mógłbyś taki czujnik zastosować.

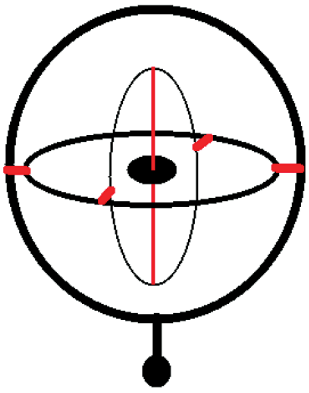
5.2.3. Czujnik dotykowy

Postaraj się wykonać takie same czynności, jak w poprzednich punktach dotyczących czujnika dotyku. Ten czujnik jest kontaktowy – czerwony fragment czujnika (zobacz ilustracja 5.2) musi być wciśnięty, aby czujnik wykrył kontakt. Czy potrafisz podać sytuację, w której taki czujnik znajdzie zastosowanie?

5.2.4. Żyroskop

Czujnik ten wykrywa orientację i ewentualny obrót. Na początku omówimy działanie żyroskopu mechanicznego. Jest to urządzenie mechaniczne składające się z wirującego z dużą prędkością dysku lub obręczy. Ten obiekt jest zawieszony na trzech obręczach umożliwiających swobodny obrót wokół dowolnej osi. Ta-

kie zawieszenie nazywa się zawieszeniem Cardano, a ostatnio popularne stało się nazywanie go nazwą angielską „gimbal", szczególnie w kontekście stabilizacji kamery umieszczonej na takiej konstrukcji. Schemat przedstawiono na ilustracji 5.6.



Ilustracja 5.6. Schemat żyroskopu. Osie obrotów i zarazem punkty montowania oznaczono czerwonymi liniami. Czarny okrąg w środku to wirujący z dużą prędkością dysk lub obręcz

Obręcze mogę się poruszać wzajemnie względem siebie, sprawiając, że umieszczony w środku obiekt (tu obracający się dysk) ma swobodę obrotów. Czy jednak w każdym ułożeniu taka swoboda występuje? Co się stanie, gdy wszystkie trzy obręcze ułożą się w jednej linii, a my chcemy wykonać obrót wokół tych linii? Mechanizm blokuje się. Takie zjawisko ma głęboki związek z opisem matematycznym (grupy) obrotów, którego ze względu na złożoność nie będziemy tu przytaczać[4].

Zastanówmy się, jakie zjawisko fizyczne sprawia, że taki żyroskop (po rozkręceniu) utrzymuje stały kierunek w przestrzeni. Za to zjawisko odpowiedzialne jest zachowanie momentu pędu. Gdy rozkręcimy dysk, ze względu na bardzo małe tarcie podczas obrotów, zachowany jest jego moment pędu. Moment pędu jest wektorem[5] skierowanym wzdłuż osi obrotu dysku. Ponieważ zawieszenie Cardano pozwala na swobodny obrót dysku, w przestrzeni kierunek momentu pędu się nie zmienia. To jest właśnie istota zachowywania stałej orientacji w przestrzeni – otoczenie się obraca, a żyroskop zachowuje orientację zadaną przez moment pędu.

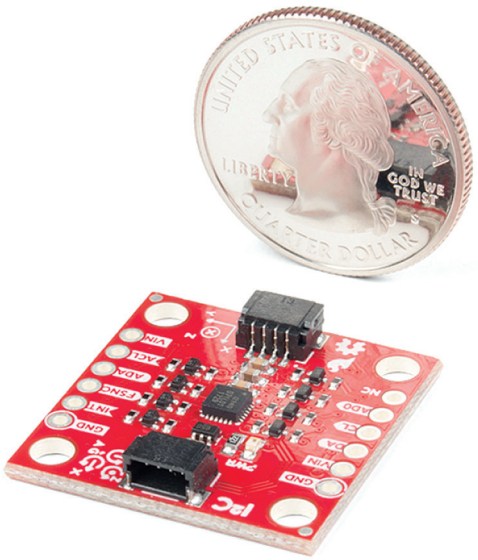
Układy żyroskopowe stosowane w elektronice, np. żyroskopy w smartfonach, są znacznie mniejsze i zintegrowane w małej obudowie. Są to urządzenia typu MEMS (ang. *microelectromechanical systems*), a skrót ten oznacza małe urządzenia mechaniczne zintegrowane z elektroniką w małym układzie. Zdjęcia takiego żyroskopu zintegrowanego z akcelerometrem (pomiar przyśpieszenia) i kompasem są przedstawione na ilustracji 5.7.

Wróćmy do naszego żyroskopu, spróbuj powtórzyć kroki podobne do tych, które wykonałeś podczas analizy dzia-

[4] Jeżeli jesteś zainteresowany to wyszukaj w Internecie hasło: *gimbal lock*.

[5] Osoby znające lepiej zaawansowaną matematykę wiedzą, że przypomina on wektor, ale przy odbiciach przestrzennych zachowuje się inaczej. Jest on czymś, co możemy traktować jako „zorientowana płaszczyzna" lub „biwektor", chociaż fizycy nazywają taki typ obiektu wektorem osiowym/aksjalnum, czyli wektorem, który jednak nie jest wektorem. W prostych zastosowaniach wystarczy, że potraktujemy ten obiekt jako zwykły wektor.

łania poprzednich czujników. Do czego mógłbyś użyć takiego żyroskopu?



Ilustracja 5.7. Przykładowy układ MEMS – układ SparkFun MPU-9250 9DoF. Są to 3-osiowy akcelerometr, żyroskop i magnetometr. Czujnik ten możemy podłączyć do miniaturowego komputera, tworząc urządzenie przenośne (materiały producenta [6])

Ludzki organizm również potrzebuje „urządzenia", które pozwala zachować orientację. Taki narząd, zwany błędnikiem, znajduje się w uchu środkowym. Jego uszkodzenie powoduje problemy z zachowaniem równowagi i orientacją. Być może to zainspiruje cię do wykorzystania żyroskopu w robotyce.

5.3. Podsumowanie

Znasz już wszystkie istotne elementy, które umożliwiają konstrukcję robotów i skomplikowanych mechanizmów. W następnym rozdziale połączymy wszystkie przekazane do tej pory wiadomości, aby zastosować je w praktyce i zbudować ciekawe konstrukcje.

Rozdział 6

Robotyka

W tym rozdziale w końcu skonstruujesz prawdziwego robota. Nauczysz się, jak połączyć wszystkie informacje uzyskane do tej pory, aby efektywnie tworzyć własne konstrukcje.

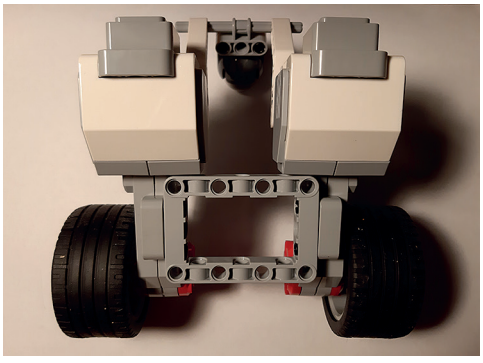
Słowo „robot" pojawiło się po raz pierwszy w czeskiej powieści fantastycznonaukowej[6] *R.U.R.* (czk. *Rossumovi Univerzální Roboti*, Uniwersalne Roboty Rossumowa) z 1921 roku, której autorem był Karel Čapek.

6.1. Robot na kołach

W tej części nauczysz się, jak wykonać platformę jeżdżącą podobnie jak samochody. Stopniowo będą dodawane nowe elementy, abyś zobaczył, jak buduje się złożone konstrukcje, startując od prostszych układów.

6.1.1. Platforma na kołach

Jako pierwszy przykład przedstawiona zostanie konstrukcja robota poruszającego się na kółkach podobnie jak samochód. Wykorzystamy do tego duże silniki. Skonstruuj platformę podobną do tej z ilustracji 6.1. Tylna część platformy w tym przypadku opiera się na kulce. Jest to najprostsze, ale nie idealne rozwiązanie, jeżeli chcesz, aby tył robota nie przeszkadzał podczas skrętów. Możesz wykorzystać dodatkowy zestaw, aby zaprojektować inny mechanizm.



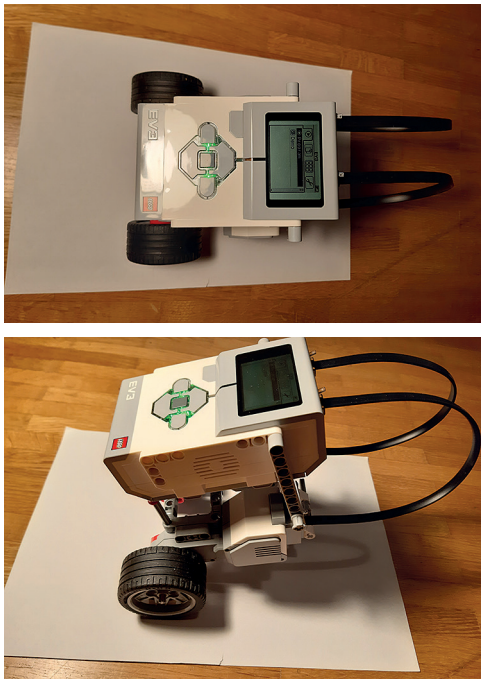
Ilustracja 6.1. Platforma jeżdżąca

W następnym kroku postaraj się w stabilny sposób osadzić kontroler, gdyż platforma będzie się poruszać. Możesz zasugerować się ilustracją 6.2.

Spróbuj teraz zaprogramować robota tak, aby poruszał się przez kilka sekund do przodu. Staraj się, żeby ten czas na początku był krótki. Pierwszy ruch jest

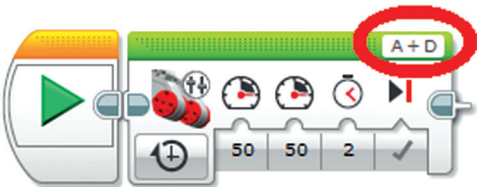
[6] Terminem angielskim jest *science fiction* lub skrót *sci-fi*.

ważny, aby określić, czy prędkość robota jest wystarczająca, żeby nie zderzył się ze ścianą lub przeszkodą w sali. Pamiętaj również, że jeżeli robot wymknie się spod kontroli podczas realizacji programu, to możesz go przerwać, naciskając przycisk wyjścia (Cancel) z ilustracji 2.9. Dzięki temu szybko zapobiegniesz buntowi maszyny.



Ilustracja 6.2. Platforma jeżdżąca z kontrolerem

Spróbuj stworzyć program przedstawiony na ilustracji 6.3. Najbezpieczniej będzie, jeżeli na początku prześlesz program, a następnie uruchomisz go, używając przycisków na kontrolerze. Pamiętaj, aby przed wysłaniem robota na spacer odłączyć kabel USB, jeżeli go używasz. Jeśli robot się nie porusza, to sprawdź, do których portów alfanumerycznych (A, B, C, D) podpiąłeś silniki, i ustaw te same porty w programie – zobacz: czerwone kółko na ilustracji 6.3 – a następnie prześlij program jeszcze raz do kontrolera i uruchom robota.



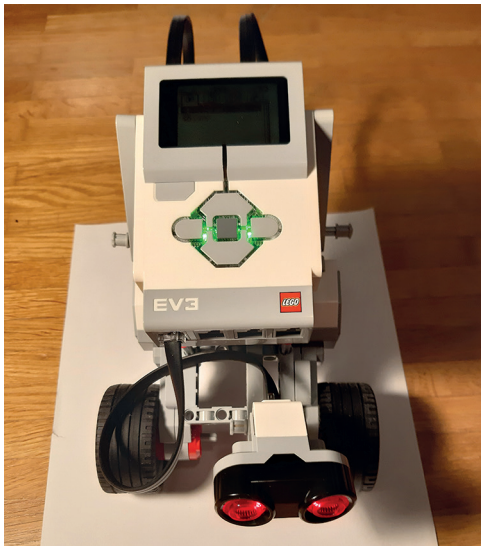
Ilustracja 6.3. Program do poruszania platformy z kołami przez 2 sekundy do przodu

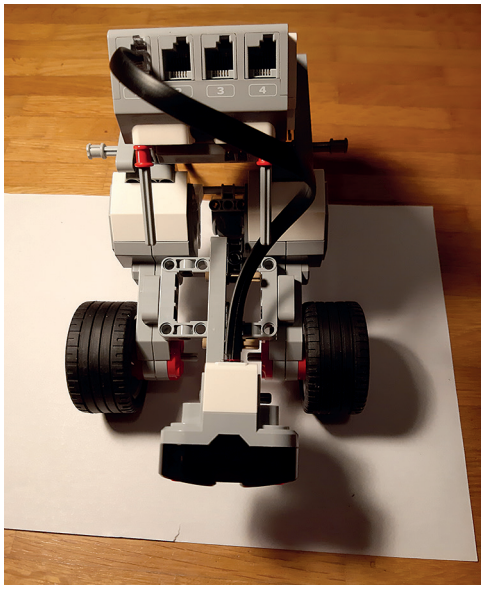
Nasz robot jedzie bezmyślnie do przodu. Niestety problemem są przeszkody. W następnym kroku rozbudujemy robota tak, aby wykrywał przeszkody i nie wjeżdżał w nie.

6.1.2. Platforma na kołach wykrywająca przeszkody

W kolejnym kroku dodamy czujnik przeszkód (ultradźwiękowy). Dzięki temu robot będzie mógł wykrywać zbliżające się przeszkody podczas poruszania się w przód. Przykładowa konstrukcja jest przedstawiona na ilustracji 6.4.

Aby zapewnić bezpieczne poruszanie się robota w obszarze z utrudnieniami, będzie on poruszał się o małą odległość do przodu, a następnie sprawdzał, czy czujnik nie zgłasza przeszkód. Jeżeli nie będzie elementów blokujących, robot będzie kontynuował ruch, a w przypadku wykrycia trudności, robot zmieni kierunek ruchu. Ważne jest to, aby robot wykrył przeszkodę, gdy będzie daleko od niej – musi mieć miejsce na wykonanie



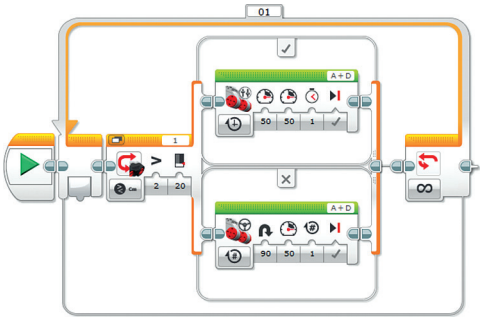


Ilustracja 6.4. Platforma na kołach z czujnikiem odległości z przodu

zwrotu. Nasz algorytm ma więc następującą postać:

- Nieskończona pętla:
  - Czy przeszkoda jest w odległości mniejszej niż np. 20 cm?
    - TAK: Wykonaj obrót o 180°;
    - NIE: Wykonaj ruch do przodu (oba silniki) przez 1 sekundę.

Spróbuj go zaimplementować. Rozwiązanie znajdziesz na ilustracji 6.5.



Ilustracja 6.5. Program dla platformy na kołach z czujnikiem odległości z przodu. Jeżeli odległość do przeszkody jest nie mniejsza niż 20 cm, to platforma jedzie do przodu – oba silniki wykonują jeden pełny obrót. Jeśli jednak czujnik ultradźwiękowy wykryje przeszkodę w odległości mniejszej niż 20 cm, wówczas platforma skręca. Podejmowanie decyzji jest otoczone nieskończoną pętlą, dzięki czemu ruch robota jest ciągły. Odpowiedni dobór parametrów programu – minimalna odległość do przeszkody, liczba obrotów przy ruchu do przodu oraz kąt skrętu po wykryciu przeszkody wpływają na płynność ruchu i responsywność robota

Jeżeli twój robot unika przeszkód, to masz dobry punkt startowy, aby spróbować dodać nowe funkcje. Możesz w przypadku wykrycia przeszkody wyświetlić na ekranie informację lub odtworzyć dźwięk. Spróbuj również dodać bardziej złożone zachowania robota – nie bój się eksperymentować.

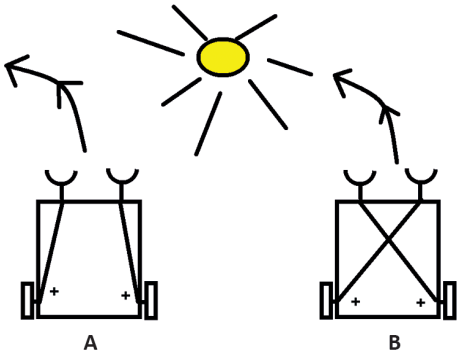
Podobnie zachowują się odkurzacze automatyczne. Spróbuj zastanowić się, jakie problemy należy rozwiązać, aby stworzyć efektywny odkurzacz, który

posprząta podłogę również blisko przeszkody. Czy pomocny byłby wówczas czujnik dotyku? Zaawansowane roboty odkurzające tworzą plan pomieszczenia, aby efektywnie zaplanować sprzątanie.

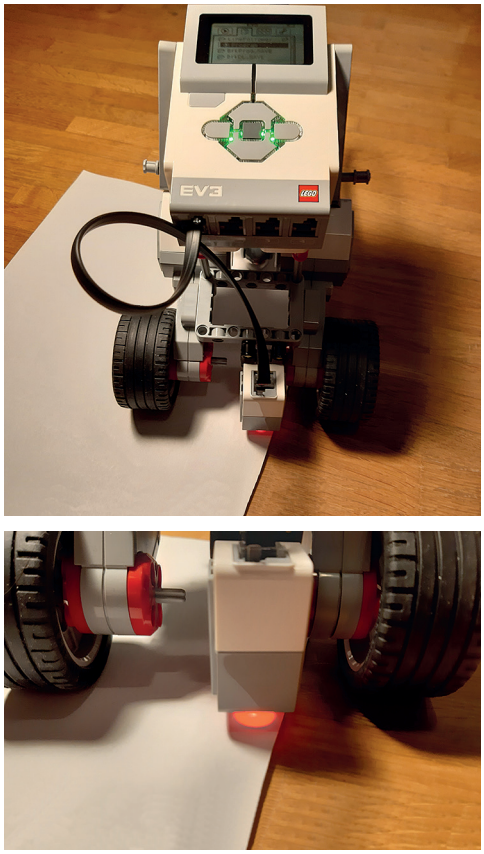
6.1.3. Platforma na kołach – podążamy za grubą linią

Teraz spróbujemy wykorzystać czujnik koloru, aby zbudować robota, który porusza się wzdłuż granicy obszarów jasnego i ciemnego. Może to być na przykład linia ułożona z pasów kartki przyklejonych na podłodze.

Taki minimalistyczny model jest przykładem tzw. pojazdu Braitenberga. Jest to eksperyment myślowy, który miał pokazywać, jak złożone zachowania są tworzone na podstawie prostszych, jak np. fototaksja, czyli zjawisko reakcji na bodźce świetlne. Schemat dwóch takich pojazdów przedstawia ilustracja 6.6. Pojazd *A* ma czujniki światła połączone z odpowiadającymi im silnikami napędzającymi koła. Symbol + oznacza, że gdy czujnik wykryje większe natężenie światła, koło będzie się szybciej obracało. Jest to przykład dodatniego sprzężenia. Zatem w przypadku tego pojazdu *A* połączenie czujników z silnikami sprawia, że będzie on stronił od źródła światła. W przypadku *B* czujnik i silnik połączone są na krzyż – zauważ, że w środku przewody się nie łączą. W takiej konfiguracji pojazd będzie poruszał się w kierunku światła. Spróbuj wyjaśnić dokładnie, jak to się dzieje.



Ilustracja 6.6. Przykłady pojazdów Braitenberga reagujących na światło. Wypustki w kształcie litery Y to czujniki światła. Połączenie czujnika z silnikiem + oznacza, że gdy czujnik wykryje duże natężenie światła, połączony z nim silnik obraca się szybciej

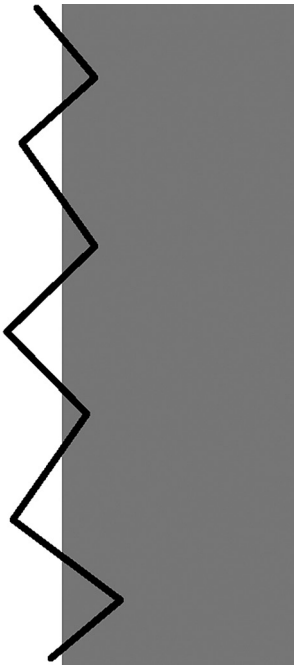


Ilustracja 6.7. Robot podążający za linią

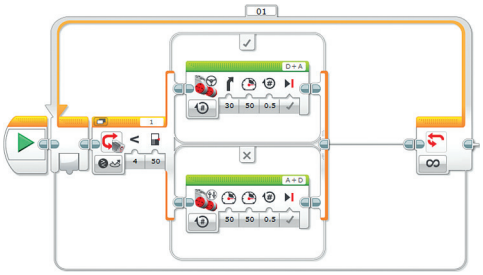
Po tym wstępie teoretycznym wróćmy do konstrukcji naszego pojazdu. Na początku zamontuj z przodu czujnik koloru na wzór przedstawionego na ilustracji 6.7. Widzimy czujnik koloru, który działa w trybie wykrywania jasności, podświetlając kartkę. Tryb jasności wystarczy nam, gdy jeden z obszarów (np. kartka) znacznie lepiej odbija światło niż drugi (podłoga). Pomysł programu polega na zachowaniu wspomnianego wyżej pojazdu Braitenberga. Jedyną różnicą jest teraz to, że będziemy kierowali się jasnością wskazywaną przez czujnik. Schemat ruchu robota na granicy dwóch obszarów przedstawiono na ilustracji 6.8. Algorytm jest bardzo prosty:

- Nieskończona pętla:
  - Czy czujnik koloru wykrył jasny obszar?
    - TAK: Skręć w kierunku obszaru ciemniejszego i poruszaj się przez krótką chwilę prosto;
    - NIE: Skręć w kierunku obszaru jasnego i poruszaj się przez krótką chwilę na wprost.

Czy już domyślasz się, jak powinien wyglądać program realizujący ruch robota przy ustawieniu obszarów na ilustracji? Spróbuj taki program napisać i przetestować. Na początek musisz skalibrować czujnik – dowiedzieć się, jaką jasność wskazuje na kartce, a jaką na podłodze. Spróbuj na kontrolerze z podłączonym czujnikiem sprawdzić, jaki odczyt daje czujnik jasności dla jasnego i ciemnego obszaru, aby tych wartości użyć w programie. Te wartości silnie zależą od tego, jak jasne są obszary. W przypadku białej kartki na jasnej podłodze różnice w jasności mogą być minimalne. Jeśli to jest problem, to możesz spróbować użyć kartki czarnej. Jeżeli masz problem z napisaniem programu, to rozpocznij od tego z ilustracji 6.9. Program ten opisuje zawracanie robota do obszaru jasnego (po lewej stronie w kierunku ruchu), gdy wykryje obszar ciemniejszy. By działał on poprawnie, należy ustawić robota pod takim kątem, aby z obszaru jasnego wjechał do ciemniejszego. Wówczas robot powinien zawrócić. Uogólnienie tego programu będzie polegało na dodaniu (krótkiego) ruchu do przodu po zmianie kierunku oraz dodaniu analogicznego bloku, w momencie gdy jesteśmy w obszarze jasnym.



Ilustracja 6.8. Poruszanie się na granicy obszarów



Ilustracja 6.9. Program do robota poruszającego się wzdłuż granicy obszarów. Program opisuje przypadek, w którym robot wjeżdża do obszaru ciemniejszego i wówczas wraca do jaśniejszego

Jeżeli uda Ci się sprawić, aby robot poruszał się wzdłuż granicy obszarów, spróbuj napisać program, który sprawi, że robot pozostaje w obszarze jasnym, który graniczy z ciemniejszym po obu stronach. To będzie symulowało ruch autonomicznego samochodu wzdłuż pasa ruchu.

Jeśli zrozumiałeś istotę problemu podążania granicą dwóch obszarów, możesz spróbować swoich sił w przypadku problemu, w którym linia jest wąska. Musisz zabezpieczyć robota tak, aby nie zgubił linii i nie zaczął jeździć po okręgu.

Innym ciekawym podejściem do budowy platformy poruszającej się jest użycie zamiast kół gąsienic. Możesz je skonstruować, tak jak pokazuje to ilustracja 6.10. Gąsienic możesz też użyć jako przekładni lub pasa transmisyjnego, na którym przesuwają się produkty w fabryce. Wróć do platformy poruszającej się i spróbuj zastosować napęd jak w czołgu, który został przedstawiony na ilustracji 6.11. Zauważ, że jeśli chcesz skręcić takim pojazdem, to gąsienice muszą się kręcić w przeciwnych kierunkach.

Ilustracja 6.10. Przykład konstrukcji gąsienicy

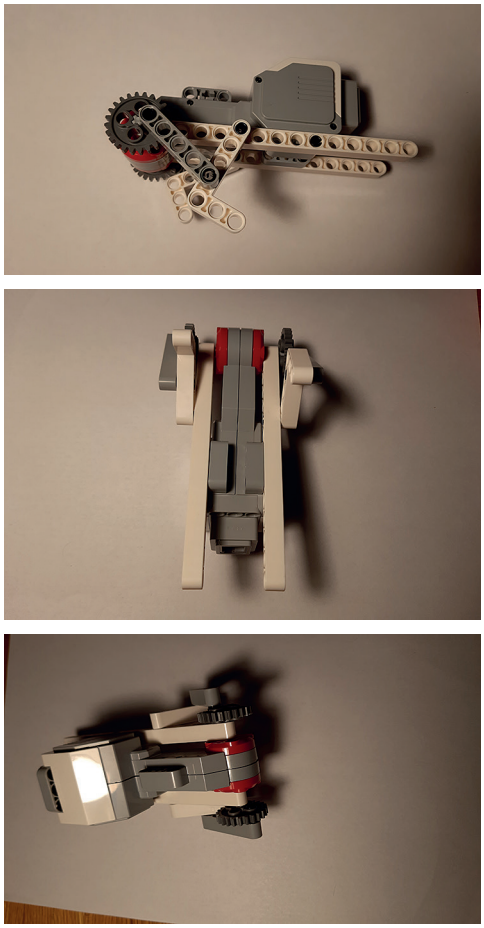
Ilustracja 6.11. Czołg – widoczne są gąsienice. Gąsienice, zamiast kół, znacznie lepiej sprawdzają się w trudnym terenie [7]

6.2. Robot kroczący

Odmianą robota na kółkach będzie robot, który porusza się w sposób przypominający ruch nóg człowieka. Do stworzenia takiego robota będzie potrzebna wiedza zdobyta podczas omawiania maszyn prostych i przekładni. Podobnie jak poprzednio wykonuj instrukcję powoli, aby mieć pełną kontrolę nad procesem tworzenia i testować stworzone konstrukcje na każdym etapie, jak robią najlepsi programiści i konstruktorzy.

6.2.1. Budowa platformy kroczącej

W kroku pierwszym, używając dużego silnika, skonstruuj platformę przedstawioną na ilustracji 6.12. Jak zapewne zauważyłeś, robot będzie miał tylko przednie nogi, a jego tył będzie niezbyt zgrabnie sunął po podłodze – od czegoś musimy rozpocząć, a rozbudowa jest twoim zadaniem. Zacznij od podłączenia robota do kontrolera odpowiednio długim kablem, aby miał swobodę ruchów.

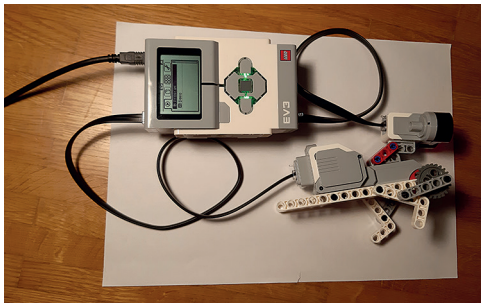


Ilustracja 6.12. Platforma krocząca

Program testowy będzie bardzo prosty – robot ma poruszać się do przodu przez pewien czas. Dzięki programowi dowiesz się, w którym kierunku robot idzie i jakie ulepszenia wprowadzić. Program jest podobny do tego z ilustracji 6.3. Jedyna różnica jest taka, że wykorzystujesz tylko jeden silnik! Zadbaj o to, aby wpisać dobrze literę portu, do którego go podłączyłeś. Jeżeli robot działa, przejdźmy do wykrywania przeszkód.

6.2.2. Wykrywanie przeszkód

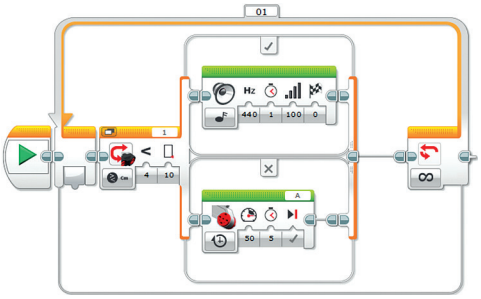
W kolejnym kroku na przetestowanej platformie kroczącej powinieneś zamontować (ultradźwiękowy) czujnik zbliżenia, jak to pokazano na ilustracji 6.13.



Ilustracja 6.13. Platforma krocząca z czujnikiem zbliżeniowym

Program bazowy będzie bardzo prosty. W przypadku wykrycia przeszkody robot wyda dźwięk. Twoim zadaniem będzie jego stopniowe ulepszanie. Spróbuj napisać taki program. Jeśli masz problemy, zobacz przykładowe rozwiązanie na ilustracji 6.14.

Spróbuj dodać tylne nogi do robota i zaprogramować go tak, aby zmieniał kierunek ruchu.

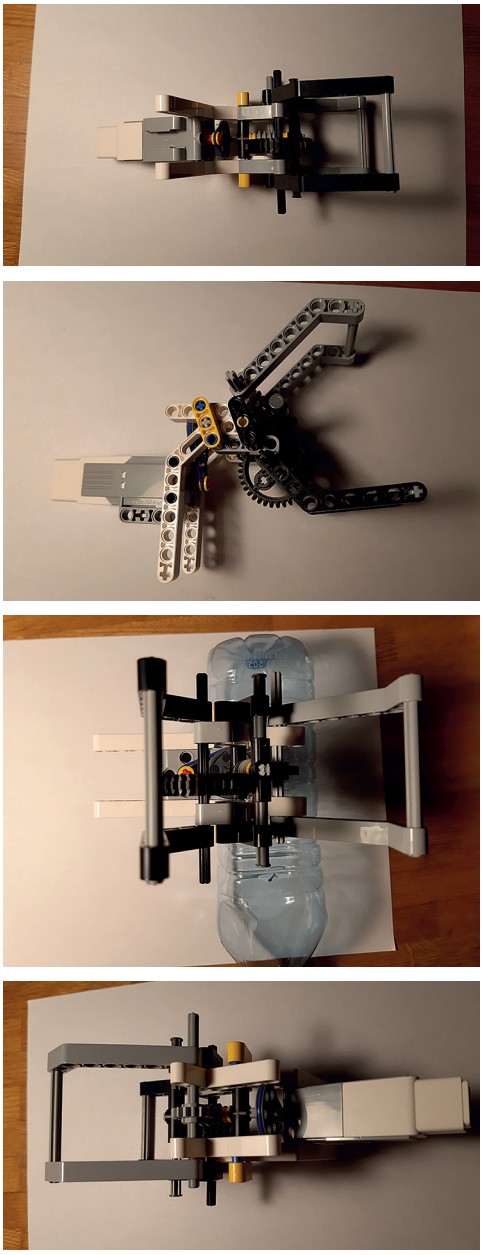


Ilustracja 6.14. Program do platformy kroczącej z wykrywaniem przeszkód

6.3. Mechanika ręki

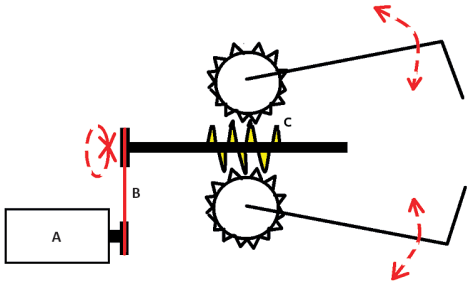
W tej części twoim zadaniem będzie skonstruowanie chwytaka, który imituje dłoń. Jest to punkt startowy do konstrukcji robotów humanoidalnych lub protez. Pamiętaj jednak, że ręka ma bardzo wiele stopni swobody, więc jest ją trudno odtworzyć w szczegółach. Dlatego nie zniechęcaj się prostotą naszej konstrukcji i potraktuj to, co stworzysz, jako punkt startowy do bardziej zaawansowanych mechanizmów.

Na początek skonstruuj rękę zgodnie z instrukcją na ilustracji 6.15. Jej schemat pokazano na szkicu 6.16. Użyj precyzyjnego silnika. Zauważ, że powinieneś użyć przekładni cięgnowej z napędem przeniesionym przy pomocy niebieskiej gumki, którą znajdziesz w zestawie rozszerzającym. Użycie przekładni ślimakowej jest tutaj uzasadnione, gdyż chcemy, aby chwytak poruszał się powoli i precyzyjnie.



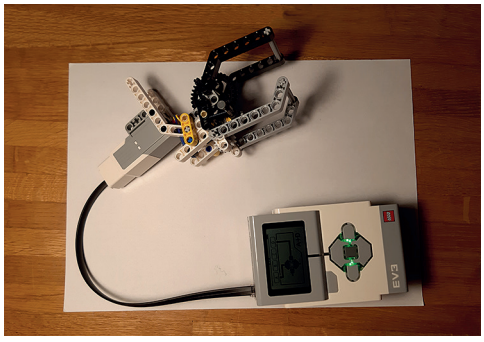
Ilustracja 6.15. Konstrukcja chwytaka

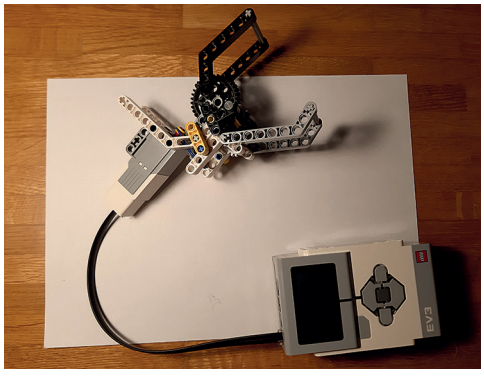
Jest to najbardziej zaawansowana konstrukcja ze wszystkich, które do tej pory wykonałeś, więc nie przejmuj się niepowodzeniami – pamiętaj, że się uczysz. Jeśli jednak udało Ci się wykonać rękę, podłącz ją teraz do kontrolera



Ilustracja 6.16. Szkic chwytaka: A – silnik; B – przekładnia cięgnowa (użyj gumki jako pasa napędowego); C – przekładnia ślimakowa napędza dwa koła zębate u góry i dołu, które poruszają „palce" chwytaka

i używając klawiszy na klocku, uruchom silnik i sprawdź, czy zaciska i otwiera się zgodnie z założeniami. Podłączenie przedstawia ilustracja 6.17.

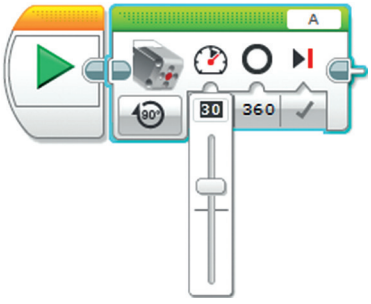




Ilustracja 6.17. Kontrola chwytaka przy pomocy klocków

Jeżeli nasza „sztuczna ręka" działa, to teraz czas z nią porozmawiać przy pomocy komputera, jak to zostało powiedziane w filmie *Terminator 3. Bunt Maszyn* – „Porozmawiajmy z dłonią".

Stworzony program będzie obracał silnikiem precyzyjnym. Na początku postaraj się, aby kąt obrotu był mały, aby sprawdzić, jaki jest zakres ruchów, i nie zniszczyć mechanizmu. Przykładowy program przedstawiono na ilustracji 6.18. Jeśli program nie działa, sprawdź, czy port, do którego jest podłączony silnik, jest taki sam, jak określony w programie. Jeśli wszystko się udało, to masz okazję, aby poeksperymentować, rozbudowując stworzoną rękę, a także program.



Ilustracja 6.18. Program do poruszania chwytakiem

6.4. Podsumowanie

W tym rozdziale poznałeś kilka typów konstrukcji mechatronicznych oraz robotów. Możesz te wiadomości wykorzystać do konstrukcji bardziej skomplikowanych mechanizmów. Ograniczeniem jest jedynie twoja wyobraźnia.

Rozdział 7

Zakończenie

Znasz już zasady tworzenia prostych robotów, a także elementy informatyki, matematyki oraz fizyki lezące u ich podstaw. Jak zauważyłeś, jest to obszar multidyscyplinarny, wymagający wiedzy we wszystkich tych dziedzinach i umiejętnego jej łączenia. Jeżeli te zajęcia ci się spodobały i chcesz rozwijać swoją karierę w proponowanym kierunku, to wiesz już, na co zwrócić uwagę na zajęciach w szkole.

Na zakończenie omówimy kilka tematów z robotyki, które mogą cię zainteresować. Jeżeli chciałbyś zobaczyć wykłady specjalistów na temat robotyki i sztucznej inteligencji, to możesz wziąć udział w Wykładach Szanghajskich z Robotyki i Sztucznej Inteligencji [12]. Mają one postać telekonferencji pomiędzy uniwersytetami na całym świecie specjalizującymi się w omawianych tematach. Z tych zajęć możesz dowiedzieć się więcej na temat obecnego stanu rozwoju tych dziedzin.

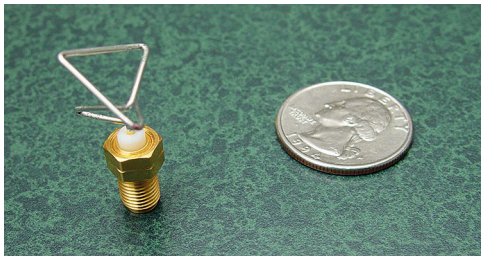
Sztuczna Inteligencja (AI – ang. *Artificial Intelligence*) jest terminem oznaczającym inteligencję posiadaną przez maszynę w odróżnieniu od ludzkiej inteligencji. Jest to dziedzina informatyki bardzo szybko rozwijająca się. Ważnym zagadnieniem jest test Turinga od nazwiska jednego z ojców nowoczesnej informatyki Alana Turinga. Polega on na odpytywaniu obiektu testu przez osobę będącą sędzią. Test sprawdza, czy obiekt potrafi posługiwać się językiem naturalnym oraz czy potrafi prowadzić konwersację w sposób autonomiczny. Ma on na celu sprawdzenie, czy uczestnik potrafi w konwersacji odróżnić komputer od człowieka. W 2014 roku komputer przeszedł test Turinga – rozmowa z programem była nieodróżnialna od rozmowy z człowiekiem. To sugeruje, że wkraczamy w okres, w którym program komputerowy potrafi prowadzić z człowiekiem konwersację. Nie należy się jeszcze obawiać ery dominacji robotów, o czym mówi popularny w Internecie cytat: „Nie boję się maszyny, która zda test Turinga. Boję się maszyny, która celowo go obleje”[7].

Ciekawym podejściem do konstrukcji robotów jest pozwolenie programowi na ich automatyczne projektowanie. Czasami pomocne są tutaj algorytmy

[7] Na podstawie angielskiego cytatu Iana McDonalda: „Any AI smart enough to pass a Turing test is smart enough to know to fail it”.

ewolucyjne, które naśladują ewolucję w świecie biologicznym według Darwina – promowane są rozwiązania, mające charakterystyki lepsze od poprzedników, a usuwane te, które są gorsze. Ta dziedzina nazywa się robotyką ewolucyjną.

Wynik działania takiej ewolucji może być nieoczekiwany, jak pokazuje przykład z użyciem algorytmu ewolucyjnego do konstrukcji anteny dla NASA. Algorytm opracował antenę o kształcie przedstawionym na ilustracji 7.1. Jej kształt dla inżynierów tworzących anteny był nieoczekiwany, a jednak okazała się ona niezwykle efektywna i została użyta w konstrukcji satelity. Szczegółowy opis problemu znajdziesz w [13]. Przykład ten pokazuje, że czasami projekt wykonany przez człowieka nie jest najbardziej optymalny. Nie zawsze jesteśmy na tyle kreatywni, aby opracować nową jakość. Algorytmy genetyczne nie mają takich ograniczeń – starają się znaleźć optymalne rozwiązanie w wyznaczonym przez nas obszarze poszukiwań.



Ilustracja 7.1. Pierwsza wykorzystana w misji kosmicznej antena opracowana przez algorytm ewolucyjny i opisana w publikacji [13]. Zdjęcie pochodzi ze zbiorów NASA

Robotyka i sztuczna inteligencja grają obecnie coraz większą rolę w przemyśle i życiu człowieka. Z raportu *Pracownik przyszłości* [14] opracowanego przez Infuture Hatalska Foresight Institute dla firmy Samsung wynika, że większość zawodów wymagających niskich kwalifikacji będzie w przyszłości podlegała automatyzacji. Jedynie zawody wymagające kreatywnego myślenia (np. naukowiec), empatii (praca z ludźmi, np. psycholog, opiekun osób starszych, wychowawca) lub skomplikowanych zdolności manualnych (np. chirurg) będą opierały się postępującej automatyzacji. Pojawią się również nowe interesujące zawody. Automatyzacja jest wyzwaniem dla współczesnych systemów społeczno-ekonomicznych opartych na pracy człowieka. Rodzi się więc ważne pytanie: Czy obecny system przetrwa, gdy roboty będą masowo wypierały ludzi z powszechnych, wymagających mniejszej specjalizacji zawodów?

Obecnie coraz częściej spotykamy autonomiczne pojazdy, które bez udziału człowieka potrafią dowieźć bezpiecznie pasażera do celu. Ich oprogramowanie wykorzystuje skomplikowane algorytmy przetwarzania obrazu połączone z metodami sztucznej inteligencji, aby śledzić sytuację na drodze. Pewną autonomię posiadają również pojazdy kosmiczne i łaziki wysyłane na inne planety, np. na Marsa. Ze względu na duże odległości w Układzie Słonecznym występują opóźnienia w sygnale radiowym rzędu minut, a nawet godzin. W takiej sytuacji pojazd manewrujący na orbicie planety, lądujący na niej lub łazik poruszający się po planecie nie mogą być pilotowane w czasie rzeczywistym z Ziemi. Taki instrument musi więc część zadań wykonywać sa-

modzielnie (autonomicznie). System taki powinien być zaprojektowany tak, aby radził sobie z awariami i różnymi nieoczekiwanymi wydarzeniami, gdyż nie ma możliwości jego naprawy.

Robotyka jest obecnie na tyle obszerną dziedziną, że trudno jest podać kompletny podział typów robotów, dlatego poniżej wskazano kilka kierunków rozwoju, które mogą cię zainteresować:

- roboty „społeczne" – posiadają umiejętności wchodzenia w szeroką interakcję z człowiekiem, mogą służyć jako asystenci w pracy i domu, do pomocy osobom starszym lub niepełnosprawnym, używane są jako punkty informacyjne oraz w terapii (ilustracja 7.2).
- roboty przemysłowe – używane w fabrykach do automatyzacji pracy; takie roboty w przyszłości będą wypierały ludzi z zawodów, które są łatwe w automatyzacji.
- roboty humanoidalne – roboty naśladujące człowieka;
- roboty biomimetyczne – roboty naśladujące organizmy żywe, np. ich ruchy, takim robotem jest CyberRyba tworzona na Politechnice Krakowskiej; dziedziny zajmujące się tego typu robotami to bionika lub biomimetyka;
- roje robotów – zbiór robotów, które potrafią się organizować w celu wykonania konkretnego zadania; przykładem może być rój dronów lub małe nanoroboty potrafiące się ze sobą komunikować;
- roboty medyczne – roboty używane w medycynie; mogą to być urządzenia umożliwiające zdalną operację człowieka z nadzorem lekarza lub bez niego;
- roboty pilotowane – są one sterowane przez człowieka, często służą jako szkielet zewnętrzny (egzoszkielet) wzmacniający ludzki organizm w terapii lub przy wykonywaniu ciężkich prac.

Ilustracja 7.2. Pepper – produkt firmy SoftBank Robotics – tu pracujący jako asystent hotelowy

Część z wymienionych robotów może działać w sposób autonomiczny lub w trybie półautonomicznym nadzorowanym przez człowieka. Niektóre z nich jednak wymagają pilotowania. W takich przypadkach rozwijane są interfejsy do wygodnego i naturalnego dla człowieka sterowania. Często opierają się ona na metodach wzbogaconej rzeczywistości (ang. *augmented reality*) i wirtualnej rze-

czywistości (ang. *virtual reality*) – operator jest osadzony w wirtualnym świecie i przy użyciu manipulatorów i gestów steruje zachowaniem robota.

Dziedzina robotyki i sztucznej inteligencji rozwija się obecnie bardzo dynamicznie i z pewnością w przyszłości będzie to niezwykle istotna gałąź przemysłu. Dzięki informacjom zawartym w tym skrypcie znasz już jej podstawy i od ciebie tylko zależy, czy będziesz kontynuował naukę i doskonalił umiejętności.

Na koniec warto przytoczyć kilka innych książek, które zawierają różne ciekawe konstrukcje mechanizmów i robotów. Są to publikacje [10, 11, 8, 9]. Szczególnie polecam pozycję [10], w której podane są konstrukcje wielu mechanizmów o różnych poziomach skomplikowania. Wykorzystując wiadomości z matematyki i fizyki, które poznałeś, będziesz w stanie przeanalizować ich działanie.

Życzę ci sukcesów w odkrywaniu tajemnic mechaniki, mechatroniki oraz robotyki.

Dodatek A

Przygotowanie zajęć – dla opiekunów

Przed przystąpieniem do zajęć upewnij się, że zestawy są kompletne i nie zawierają zepsutych elementów, gdyż grozi to niebezpieczeństwem dla uczestników. Uszkodzone lub pęknięte elementy mogę podczas użycia w ruchomym mechanizmie rozpaść się i zranić uczniów. Dlatego należy je bezwzględnie usunąć z zestawu i ewentualnie zastąpić nowymi.

Klocki Lego są zasilane bateriami lub baterią dostarczaną z zestawem. Staraj się, aby bateria zawsze była całkowicie ładowana i rozładowywana, zapewni jej to długą żywotność. Ładowanie baterii powinieneś zacząć przed zajęciami, ale nie będzie ona również wykorzystywana podczas części dotyczącej mechaniki, więc jest to dodatkowy czas na tę czynność.

Zestaw zwiera bardzo dużo małych i łatwych do zgubienia części, więc staraj się uczulić uczestników na to, aby organizując miejsce pracy, umieścili pojemnik z częściami możliwie daleko od obszaru roboczego, aby zapobiec przypadkowemu trąceniu go i rozsypaniu części. Jeżeli taki przypadek się wydarzy, uczestnik powinien przerwać pracę i posprzątać stanowisko przed jej wznowieniem. Upewnij się wówczas, czy uczestnik dokładnie posprzątał stanowisko.

Uczestników należy zapoznać z ogólną ideą zajęć i podstawowymi zasadami bezpieczeństwa i higieny pracy zawartymi w rozdziale 2. W przypadku pracy z dziećmi lub młodzieżą stosuj jak najprostszy język i unikaj tzw. przynudzania. Zadbaj także o precyzję wyrażania się.

Główną ideą jest zachęcanie uczestników do własnych przemyśleń i eksperymentowania. Nie powinieneś tłumić takich odruchów na rzecz realizacji programu. Program zajęć umieszczony w rozdziałach tej książki, a także materiałach dodatkowych jest na tyle elastyczny, że możesz w dowolnym punkcie sensownie zmienić temat, wyjaśniając zainteresowanym nowe zagadnienie.

Klocki dostarczają niemal nieograniczonych możliwości konstrukcji różnego rodzaju mechanizmów i robotów. Daj podopiecznym pole do bycia kreatywnymi i czerpania radości z ich odkrywania.

Dodatek B

Mechanika

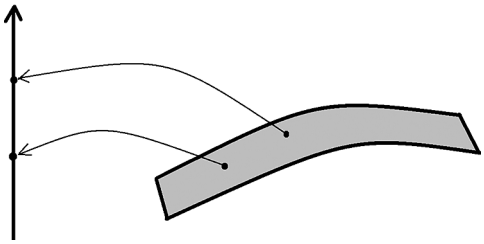
W tym rozdziale poznasz zasady fizyczne rządzące maszynami prostymi i elementarną matematykę umożliwiającą ich opis[8].

B.1. Skalar i wektor

Na początek trzeba zapoznać się z podstawowymi „cegiełkami", które pomogą zrozumieć fizyki potrzebne do rozwiązania poszczególnych problemów.

B.1.1. Wielkość skalarna

Pierwszym z tych pojęć jest wielkość skalarna (a dokładniej pole skalarne). Jest to obiekt, który każdemu punktowi w przestrzeni, w której rozpatrujemy zjawisko, przyporządkowuje liczbę (rzeczywistą). A zatem to funkcja, która „zjada" punkt i produkuje z niego liczbę z tym punktem stowarzyszoną[9], jak to przedstawia ilustracja B.1.



Ilustracja B.1. Wielkość skalarną można wyobrażać sobie jako funkcję, która pobiera punkt z pewnego obszaru i przekształca go na liczbę (rzeczywistą)

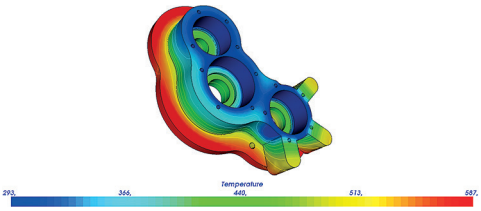
Przykładem wielkości skalarnej jest rozkład temperatury w pewnym obszarze. Na ilustracji B.2 przedstawiono symulację rozkładu temperatury w mechanicznej części. Skalary można mnożyć i dodawać do nich liczby. Dla przykładu, spróbuj stopnie Kelwina przeliczyć na stopnie Celsjusza w każdym punkcie ilustracji B.2, odejmując od wartości w Kelwinach liczbę 273.

Takimi polami skalarnymi zdefiniowanymi w czasoprzestrzeni (czas i trój-

[8] Jeśli chcesz dowiedzieć się więcej, zajrzyj do prostego podręcznika do fizyki [15], a jeśli chcesz poznać jeszcze więcej szczegółów matematycznych, sprawdź w podręczniku [16]. Pojęcia matematyczne, takie jak pochodna czy całka, znajdziesz w bardzo dobrze napisanej książce [20].

[9] Skalarem określa się też liczbę, którą można stowarzyszyć z funkcją stałą dającą tę samą liczbę dla każdego punktu w obszarze.

wymiarowa przestrzeń z odpowiednią strukturą matematyczną), w której żyjemy, opisujemy tzw. pole Higgsa, odpowiedzialne za nadawanie masy (niektórym) cząstkom elementarnym. Wzbudzenie tego pola nazywane bozonem Higgsa zostało odkryte w 2012 roku.



Ilustracja B.2. Rozkład temperatury (w stopniach Kelwina) w części silnika. Symulację wykonano w programie ElmerCSC [22]. Program ten jest bezpłatny i służy do rozwiązywania równań opisujących zjawiska fizyczne zlokalizowane w pewnych obszarach – w tym przypadku chodzi o równanie przewodnictwa cieplnego we fragmencie silnika

B.1.2. Wektor

Odrobinę bardziej złożonym obiektem jest wektor. Wektor wyobrażamy sobie jako strzałkę, która wskazuje pewien zwrot oraz ma określoną długość, mówiącą, jak „długi" jest wektor. Wektory oznaczamy literą ze strzałką, np. $\vec{a}$, jednak często nie piszemy strzałki nad wektorem. Formalnie ta wielkość składa się z czterech elementów:

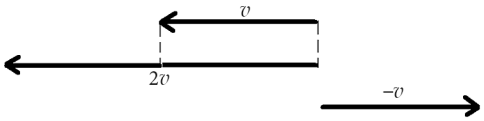
- kierunku – prosta, na której leży (i którą definiuje) wektor;
- zwrotu – jedna z dwóch możliwości, którą wskazuje strzałka wektora;
- wartości – długość wektora, którą oznaczamy $|\vec{a}|$;
- punktu zaczepienia – punkt, w którym wektor jest zaczepiony[10].

Poglądowe wyobrażenie wektora jest przedstawione na ilustracji B.3.

Ilustracja B.3. Wektor („strzałka") z kierunkiem (prosta przerywana), na którym leży. Zaznaczono też punkt zaczepienia wektora

Jednym ze sposobów zapamiętania różnicy pomiędzy kierunkiem i zwrotem jest to, że gdy pytamy, czy idziemy w dobrym kierunku, to warto dopytać, czy również zwrot jest właściwy. To szczególnie ważne w przypadku, gdy pytamy o drogę matematycznego purystę.

Wektory możemy mnożyć przez liczby (rzeczywiste), odpowiednio wydłużając lub skracając ich długość. Dodatkowo, jeżeli wektor pomnożymy przez liczbę ujemną, to jego zwrot zmieni się na przeciwny (ilustracja B.4).



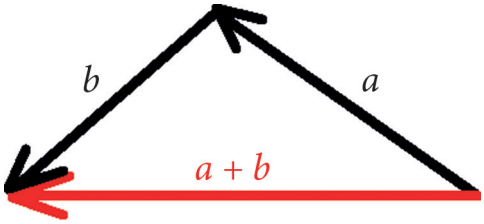
Ilustracja B.4. Wektor $\vec{v}$ (góra); ten sam wektor pomnożony przez liczbę 2, tj. wektor $2\vec{v}$ (środek); wektor pomnożony przez liczbę ujemną -1, tj. wektor $-\vec{v}$ (dół), który ma przeciwny zwrot

[10] Czasami w definicji nie podaje się punktu zaczepienia, jednak w ogólnym przypadku przestrzeni wektor musi być zaczepiony w konkretnym punkcie. Jeżeli na przestrzeni istnieje specjalna struktura matematyczna (zwana przez matematyków koneksją), umożliwiająca „transport równoległy" wektorów do dowolnego punktu, to wówczas możemy mówić o tzw. wektorach swobodnych, czyli niezaczepionych.

Jeżeli wektory mamy zdefiniowane (przypięte) w każdym punkcie obszaru, który rozważamy, to mówimy o polu wektorowym. Takie pole możemy traktować jako funkcje, które każdemu punktowi rozważanego obszaru przyporządkowują wektor w nim przypięty. Przykładem takiego pola wektorowego jest wiatr na powierzchni Ziemi[11].

Wektory od skalarów odróżnia istotnie sposób dodawania. Dwie wielkości skalarne możemy dodawać, w taki sposób, że w każdym punkcie dodajemy ich wartości – to jest wartość wielkości skalarnej będącej sumą dwóch wielkości skalarnych w punkcie. Wektory dodajemy z uwzględnieniem kierunków i zwrotów, niosą więc one więcej informacji niż wielkości skalarne. Dodawanie dwóch wektorów możemy zobrazować tak, że do końca jednego wektora doczepiamy początek drugiego wektora przesuniętego równolegle, a następnie łączymy początek pierwszego wektora z końcem drugiego, jak na ilustracji B.5. Łatwo sprawdzić, że takie dodawanie wektorów jest przemienne, wykonując dodawanie $\vec{a}+\vec{b}$, a następnie $\vec{b}+\vec{a}$ i porównując oba sposoby – wynikowy wektor powinien być taki sam w obu przypadkach.

Jak w takim wypadku odjąć dwa wektory? Wystarczy skorzystać z dodawania



Ilustracja B.5. Dodawanie dwóch wektorów $\vec{a}$ oraz $\vec{b}$ daje wektor będący sumą $\overrightarrow{a+b}=\vec{a}+\vec{b}$

i mnożenia przez skalar: $\vec{a}-\vec{b}=\vec{a}+(-\vec{b})$. Musimy zatem do wektora $\vec{a}$ dodać wektor $-\vec{b}$ o przeciwnym zwrocie niż wektor $\vec{b}$.

Ta własność pozwala nam wyróżnić wielkości wektorowe. Dla przykładu, żeby pokazać, że prędkość jest wielkością wektorową, rozważmy człowieka przemieszczającego się w jadącym pociągu. Jego prędkość (z uwzględnieniem kierunku i zwrotu) jest dana przez sumę wektorów prędkości pociągu oraz prędkości tego człowieka względem wnętrza pociągu.

Każdy z obiektów, a w tym wektor, zachowuje się w charakterystyczny sposób podczas transformacji (np. obrotów). To pozwala te wielkości poprawnie zidentyfikować i przyporządkować do obiektów fizycznych, jak np. cząstki elementarne. Kwestie związane z tym tematem prowadzą zbyt daleko poza klasyczną szkolną mechanikę. Pamiętaj jednak, że aby określić, czy dana wielkość jest wielkością skalarną czy wektorową, należy sprawdzić, czy posiada własności, które tę wielkość charakteryzują na poziomie matematyki.

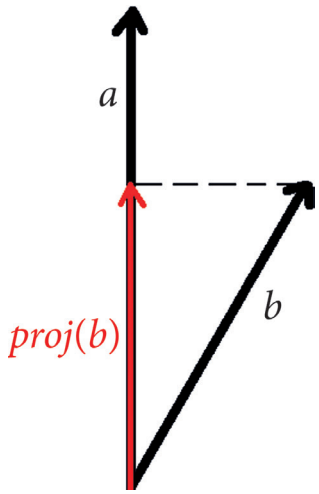
Na koniec przypomnij sobie dwie operacje związane z mnożeniem wekto-

[11] Ciekawym aspektem takiego pola wektorowego na sferze dwuwymiarowej jest to, że istnieje co najmniej jeden punkty, w którym takie pole wektorowe w kierunku stycznym do sfery znika. Istnieje słynne twierdzenie o zaczesywaniu sfery (jeża). Mówi ono, że kula z włosami nie daje się zaczesać na „gładko", co tłumaczy problemy wielu ludzi z poranną toaletą.

rów. Pierwszą z nich jest iloczyn skalarny, stowarzyszony z rzutem jednego wektora na drugi, tzn.

$$\vec{a}\vec{b} = \left|\vec{a}\right|\left|\vec{b}\right|\cos(\angle(\vec{a},\vec{b})), \qquad \text{(B.1)}$$

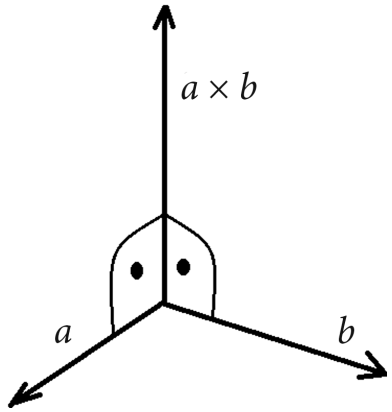
gdzie $\angle(\vec{a},\vec{b})$ jest kątem skierowanym (liczonym przeciwnie do ruchu wskazówek zegara) pomiędzy wektorami $\vec{a}$ oraz $\vec{b}$ (ilustracja B.6). Ponieważ długość projekcji wektora $\vec{b}$ ma kierunek wyznaczony przez wektor $\vec{a}$, to $\left|\vec{b}\right|\cos(\angle(\vec{a},\vec{b}))$, z czego wynika inna definicja iloczynu skalarnego wektorów. Ta interpretacja wskazuje, że iloczyn skalarny może określać, jaką składową ma jeden wektor w kierunku drugiego. Możliwość określenia takiej składowej w wybranym kierunku jest użyteczna w fizyce.



Ilustracja B.6. Iloczyn skalarny to iloczyn długości jednego z wektorów i długości projekcji wektora $\vec{b}$ na kierunek wyznaczony przez wektor $\vec{a}$ – $proj(\vec{b}) = \left|\vec{b}\right|\cos(\angle(\vec{a},\vec{b}))$

Drugim użytecznym iloczynem jest iloczyn wektorowy oznaczany przez $\times$ (ilustracja B.7). Zwrot wektora $\vec{a}\times\vec{b}$ określić można regułą prawej dłoni. Natomiast długość iloczynu skalarnego to

$$\left|\vec{a}\times\vec{b}\right| = \left|\vec{a}\right|\left|\vec{b}\right|\sin(\angle(\vec{a},\vec{b})). \qquad \text{(B.2)}$$



Ilustracja B.7. Iloczyn wektorowy wektorów $\vec{a}$ oraz $\vec{b}$

Czy iloczyn wektorowy jest wektorem? Ma on podobne właściwości do wektora, ale jeżeli spojrzymy na jego odbicie, które zmienia kierunek wektora na przeciwny, to iloczyn wektorowy nie zmieni się, gdyż

$$(-\vec{a})\times(-\vec{b}) = (-1)^2\,\vec{a}\times\vec{b} = \vec{a}\times\vec{b}. \qquad \text{(B.3)}$$

Czasami taki obiekt dla rozróżnienia nazywa się wektorem aksjalnym lub pseudowektorem (czyli „wektorem, który nie jest wektorem"). Jednak jest to w rzeczywistości inny obiekt, przedstawiający zorientowaną płaszczyznę, w której leżą wektory będące czynnikami iloczynu wektorowego. Zauważ, że tylko w przestrzeni o trzech wymiarach każdej płaszczyźnie możemy przyporządkować wektor, który jest do niej prostopadły – jest to wspomniany wektor aksjalny. Ani w dwóch (brak dostatecznej liczby

wymiarów), ani w większej niż trzy liczbie wymiarów (zbyt dużo opcji wyboru wektora dualnego do płaszczyzny) taki jednoznaczny[12] iloczyn nie istnieje. To, co w każdym wymiarze istnieje, jest płaszczyzną skierowaną, rozpiętą na tych wektorach[13].

Omówionymi powyżej obiektami zajmiemy się, badając je pod kątem użyteczności w mechanice, w kolejnych częściach tego dodatku.

B.2. Zasady dynamiki Newtona

W tym miejscu należy przypomnieć zasad dynamiki Newtona, które opisują układy mechaniczne, z którymi mamy do czynienia[14].

Na początek przypomnijmy sobie trzy prawa dynamiki Newtona, które opisują ruch punktu materialnego – ciała, którego rozmiary/kształt możemy pominąć, zajmując się opisywaniem jego ruchu.

Zasada 1. Pierwsza zasada dynamiki Newtona

W inercjalnym układzie odniesienia, jeśli na ciało nie działa żadna siła lub siły działające równoważą się, to ciało pozostaje w spoczynku lub porusza się ruchem jednostajnym (ze stałą prędkością) po linii prostej.

Układ inercjalny to taki, w którym nie występują (pozorne) siły bezwładności. Może to być układ nieprzyśpieszający lub swobodnie spadający w polu grawitacyjnym. Spoczynek lub ruch jednostajny po linii prostej jest więc ruchem swobodnym.

Z kontrapozycji[15] wynika prawo, że jeżeli w danym układzie widzimy ciało, które nie spoczywa ani nie porusza się ruchem jednostajnym po linii prostej, mamy dwie możliwości – nie jesteśmy w układzie inercjalnym lub siły działające na ciało nie równoważą się.

---

[12] W siedmiu wymiarach przestrzennych istnieje kilka możliwości definicji iloczynu wektorowego.

[13] Pojęcie płaszczyzny skierowanej w połączeniu z iloczynem skalarnych wektorów stanowi punkt wyjścia do definicji algebr Clifforda tłumaczących zachowania cząstek elementarnych związanych z ich spinem – dodatkowym stopniem swobody takich cząstek przyjmującym wartości dyskretne.

[14] Zasady dynamiki Newtona są jedynie przybliżeniem reguł rządzących ruchem w przypadku prędkości obiektów znacznie mniejszych niż prędkość światła $c = 299792458\ \frac{m}{s}$. Taki uogólniony opis został podany przez Alberta Einsteina w szczególnej teorii względności. Opis grawitacji w fizyce klasycznej jest również przybliżony i słuszny dla słabych pól grawitacyjnych w stosunku do ogólnego opisu grawitacji również podanego przez Alberta Einsteina i nazwanego ogólną teorią względności. Nie oznacza to jednak, że inżynier może ograniczyć się tylko do teorii klasycznych. Obecny rozwój technologii sprawia, że wkracza ona również w obszary, w których musimy stosować szczególną i ogólną teorię względności. Znakomitym przykładem jest system GPS służący do wyznaczania pozycji w każdym miejscu na Ziemi. Bez poprawek wynikających ze wspomnianych teorii Einsteina system ten nie byłby w stanie działać poprawnie [3, 4].

[15] Kontrapozycją implikacji $p \Rightarrow q$ (czytamy: jeżeli zachodzi p, to zachodzi q) jest stwierdzenie $\neg q \Rightarrow \neg p$ Te dwie implikacje są równoważne,

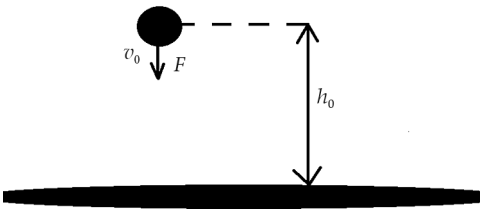
Przejdźmy teraz do drugiej zasady dynamiki, która jest użyteczna ze względu na to, że pozwala obliczać ruch ciała w przypadku, gdy na ciało działają siły.

Zasada 2. Druga zasada dynamiki Newtona

W inercjalnym układzie odniesienia, jeżeli wypadkowa sił działających na ciało jest niezerowa $\vec{F}$*, to przyśpieszenie* $\vec{a}$ *i masa ciała m są powiązane zależnością*

$$m\vec{a} = \vec{F}. \tag{B.4}$$

Zasada ta pozwala obliczyć wektor przyśpieszenia, gdy znamy układ sił działających na ciało lub vice versa. Na pewno wiesz, jak obliczyć przyśpieszenie i prędkość ciała spadającego w polu grawitacyjnym z wysokości h_0, rzuconego z prędkością początkową v_0 w dół (ilustracja B.8).



Ilustracja B.8. Ciało o masie *m* rzucone z wysokości h_0 z prędkością początkową v_0 w dół. Siła działająca na ciało od pola grawitacyjnego jest skierowana w dół i ma wartość $|\vec{F}| = mg$

Tuż przy Ziemi siła grawitacji działająca na ciało o masie *m* ma wielkość $|\vec{F}| = mg$, gdzie *g* jest przyśpieszeniem ziemskim przy powierzchni Ziemi. Wówczas prędkość ciała spadającego w czasie *t* to

$$v = v_0 + gt, \tag{B.5}$$

gdzie v_0 to prędkość początkowa ciała skierowana w dół. Natomiast położenie ciała w chwili czasu *t* to

$$h = h_0 - v_0 t - \frac{1}{2} gt^2, \tag{B.6}$$

gdzie h_0 jest wysokością początkową, z której ciało zostało rzucone w dół.

Ostatnia, trzecia zasada dynamiki mówi nam o wzajemności oddziaływania:

Zasada 3. Trzecia zasada dynamiki Newtona

Oddziaływania ciał są zawsze wzajemne. W inercjalnym układzie odniesienia siły wzajemnego oddziaływania dwóch ciał mają takie same wartości, taki sam kierunek, przeciwne zwroty i różne punkty przyłożenia.

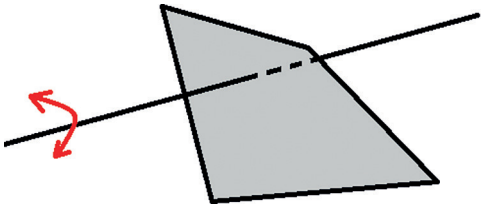
Zasadę tę możemy łatwo sprawdzić – jeżeli delikatnie pchniemy masywny obiekt, np. ciężką szafę, to również my poczujemy siłę, która nas od tego obiektu odpycha.

W tym miejscu należy przypomnieć, że jeżeli suma sił zewnętrznych działających na układ znika, to wielkość nazywana pędem i zdefiniowana jako $\vec{p} = m\vec{v}$ jest zachowana dla całego układu jako suma pędów części składowych.

a więc wyrażają to samo. Tutaj $\neg p$ jest negacją stwierdzenia *p*. Spróbuj podać zdanie prawdziwe w postaci „jeżeli *p*, to *q*", a następnie sprawdź, czy jego kontrapozycja „jeżeli nieprawda, że *q*, to nieprawda, że *p*" jest prawdziwa. Takimi pojęciami operuje dziedzina nazywana logiką (matematyczną).

Jeżeli jednak ciała są rozciągłe, jak to bywa najczęściej w rzeczywistym świecie, to musimy brać pod uwagę rozkład masy ciała. Najprostszym podejściem do opisu takiego ciała jest założenie, że ciało jest bryłą sztywną, tzn. taką, która się nie deformuje pod wpływem sił przyłożonych do różnych punktów takiej bryły. Jest to oczywiście idealizacja, a pod działaniem odpowiednio dużej siły każde ciało musi się w końcu zdeformować. Konstruując jednak układy mechaniczne, które powtarzają swój ruch wielokrotnie, należy zadbać o to, żeby siły nie były zbyt duże, aby nie doprowadzić do deformacji elementów maszyny.

Taka bryła sztywna może obracać się wokół wybranej osi (ilustracja B.9). Ruch każdej bryły możemy opisać jako złożenie ruchu postępowego oraz obrotowego bryły wokół osi. Do opisania ruchu postępowego potrzebne są prawa Newtona. Poniżej opisano ruch takiego obracającego się ciała, korzystając z powyższych zasad dynamiki ruchu obrotowego.



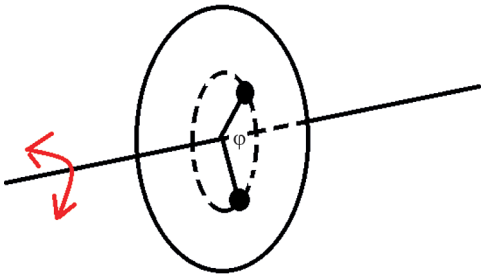
Ilustracja B.9. Bryła sztywna zaczepiona tak, aby mogła swobodnie poruszać się wokół osi

Na początku musimy podać wielkości analogiczne do prędkości, przyśpieszenia, masy i siły dla obrotu bryły sztywnej wokół osi. Ustalmy więc oś obrotu takiej bryły sztywnej. Wówczas prędkość obrotu opisuje prędkość kątowa (ilustracja B.10), którą oznacza się grecką literą ω. Jest to szybkość zmian kąta obrotu wybranego punktu ciała (poza osią obrotu) w czasie, formalnie:

$$\left|\vec{\omega}\right| = \lim_{\Delta t \to 0} \frac{\Delta\phi}{\Delta t}, \qquad \text{(B.7)}$$

gdzie $\Delta\phi$ jest zmianą kąta w czasie Δt. Operacja $\lim_{\Delta t \to 0}$ oznacza, że interesuje nas zmiana kąta w bardzo małych odcinkach czasu, a dokładniej w granicy (*limes* po łacinie oznacza granicę) odcinków czasu zmierzających do zera[16]. Prędkość kątowa jest pseudowektorem, a jej zwrot jest skierowany wzdłuż osi obrotu. Prędkość poruszania się tego wybranego punktu, który jest oddalony o $r = \left|\vec{r}\right|$ od osi obrotu, wynosi

$$\vec{v} = \vec{r} \times \vec{\omega}. \qquad \text{(B.8)}$$



Ilustracja B.10. Obrót wybranego punktu ciała w odległości r od osi obrotu o kąt ϕ

Zauważ, że im dalej punkt jest od osi obrotu (większy promień r), tym szybciej się on porusza. Z drugiej strony jego prędkość kątowa jest taka sama jak inne-

[16] Jeżeli ruch odbywa się „gładko", to taka granica nie jest zerem. Ścisła definicja takiego procesu granicznego jest omawiana na kursie Analizy matematycznej na studiach. Osoby, które są tym zainteresowane, odsyłam do książki [20].

go punktu bryły. Zapamiętaj te zależności, gdyż wiedza ta będzie ci potrzebna przy omawianiu przekładni.

Podobnie definiujemy przyśpieszenie kątowe, które jest szybkością zmiany prędkości kątowej. Poniżej oznaczone zostanie grecką literą α.

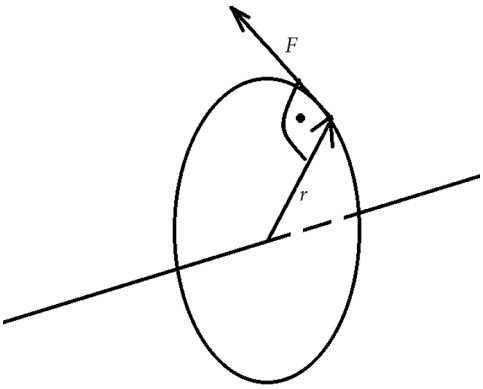
Odpowiednikiem siły jest moment siły $\vec{M}$, który opisuje siłę działającą na ciało w stosunku do wybranej osi obrotu. Jeżeli r jest odległością punktu zaczepienia siły od osi obrotu, a siła jest skierowana pod kątem prostym do osi obrotu i zgodna z obrotem (ilustracja B.11), to wówczas wartość momentu siły możemy określić w prosty sposób

$$\left|\vec{M}\right| = r\left|\vec{F}\right|. \qquad \text{(B.9)}$$

W ogólnym przypadku jest to wzór

$$\vec{M} = \vec{r} \times \vec{F}, \qquad \text{(B.10)}$$

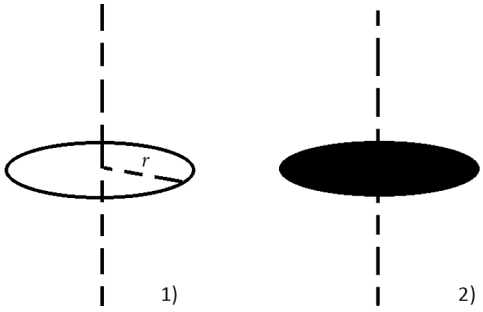
gdzie $\times$ jest iloczynem wektorowym wektorów.



Ilustracja B.11. Siła $\vec{F}$ przyłożona w odległości $\vec{r}$ od osi obrotu generuje moment pędu $\vec{M} = \vec{r} \times \vec{F}$. Ponieważ siła jest prostopadła do wektora $\vec{r}$, wielkość jej momentu możemy obliczyć z prostego wzoru $\left|\vec{M}\right| = \left|\vec{r}\right|\left|\vec{F}\right| = r\left|\vec{F}\right|$

Ostatnią wielkością jest moment bezwładności I, który jest odpowiednikiem masy w ruchu postępowym. Dla punkt materialnego o masie m obracającego się wokół osi obrotu w odległości r moment bezwładności wynosi $I = mr^2$. Dla bardziej skomplikowanych brył potrzebna jest procedura podziału bryły na punkty materialne o małej masie i zsumowanie wkładów od każdego z tych punktów. Należy to przeprowadzić w granicy mas tych punktów zmierzającej do zera. Taką procedurę realizuje się przy pomocy operacji matematycznej zwanej całką[17]. W najprostszym przypadku wykorzystuje się wzory na momenty bezwładności, np.:

- obręcz o masie m i promieniu r, dla której oś obrotu przechodzi przez środek obręczy, ma moment bezwładności $I = mr^2$;
- dysk o masie m i promieniu r, dla którego oś obrotu przechodzi przez środek dysku, ma moment bezwładności $I = \frac{1}{2}mr^2$.



Ilustracja B.12. 1) Obręcz o promieniu r i masie m; 2) dysk o takim samym promieniu i masie

[17] Po więcej informacji na temat całek odsyłam do książki [20].

Powyższe przykłady przedstawiono na ilustracji B.12. Widzimy więc, że zmiana rozkładu masy przy takich samych parametrach (r i m) może skutkować innym momentem bezwładności.

Przejdźmy teraz do zasad dynamiki dla ruchu obrotowego.

Zasada 4. Pierwsza zasada dynamiki ruchu obrotowego

Jeżeli na bryłę sztywną nie działają żadne momenty sił, to bryła ta pozostaje nieruchoma lub wykonuje ruch obrotowy jednostajny (ze stałą prędkością kątową).

Widzimy więc, że zasada ta jest analogiczna do pierwszej zasady Newtona. Czy potrafisz sformułować w analogii do zasady Newtona drugą zasadę ruchu obrotowego? Ma ona postać:

Zasada 5. Druga zasada dynamiki ruchu obrotowego

Dla ustalonej osi obrotu i niezerowego momentu siły $\vec{M}$ działającego na bryłę sztywną, związek przyśpieszenia kątowego $\vec{\alpha}$ (tj. szybkości zmian prędkości kątowej) oraz momentu bezwładności I jest następujący:

$$I\vec{\alpha} = \vec{M}. \tag{B.11}$$

Podobnie jest z trzecią zasadą dynamiki ruchu obrotowego – jest to prawo akcji i reakcji dla ruchu obrotowego.

Na koniec przypomnijmy prawo zachowania momentu pędu. Dla punktu materialnego obracającego się z pędem $\vec{p}$ w odległości $\vec{r}$ od osi obrotu moment pędu to

$$\vec{L} = \vec{r} \times \vec{p}. \tag{B.12}$$

Jak poprzednio, jeżeli kąt pomiędzy wektorem $\vec{r}$ a $\vec{p}$ jest prosty, to długość tego wektora jest zadana prostym wzorem

$$\left|\vec{L}\right| = \left|\vec{r}\right|\left|\vec{p}\right|. \tag{B.13}$$

W ogólnym przypadku dla bryły sztywnej o momencie bezwładności I i prędkości kątowej $\vec{\omega}$ moment pędu to

$$\vec{L} = I\vec{\omega}. \tag{B.14}$$

Zasada zachowania momentu pędu mówi, że jeżeli momenty sił zewnętrznych działających na układ znikają, to układ zachowuje moment pędu. Zatem (pseudo)wektor momentu pędu ma stałą wartość i kierunek, który określa też kierunek obrotu ciała. Tę własność wykorzystuje się przy konstrukcji żyroskopu.

Ostatnimi pojęciami, które będą istotne podczas zajęć, są energia i praca. Energia opisuje zdolność układu do wykonania pracy. Jest to niezwykle użyteczna wielkość, gdyż zazwyczaj chcemy wiedzieć, jak dużą pracę układ może dla nas wykonać lub ile użytecznej pracy możemy w nim zmagazynować do późniejszego wykorzystania. W zależności od rodzaju oddziaływań wyróżniamy różne rodzaje energii. Dla mechaniki istotna jest energia kinetyczna, która dla cząstki o masie m i prędkości v ma wartość

$$E_k = \frac{mv^2}{2}. \tag{B.15}$$

Jest ona związana z ruchem punktu materialnego i określa, jaką pracę wykonamy, rozpędzając lub spowalniając ciało.

Dla pola grawitacyjnego bardzo blisko powierzchni Ziemi możemy zdefiniować energię potencjalną wzorem:

$$E_p = mgh, \qquad \text{(B.16)}$$

gdzie g to przyśpieszenie grawitacyjne, a h jest wysokością ciała ponad wybrany poziom odniesienia, dla którego energia potencjalna znika. Energia ta opisuje pracę wykonaną przy unoszeniu punktu materialnego na wysokość h względem poziomu zerowego energii.

Dla każdego rodzaju tzw. sił potencjalnych możemy zdefiniować również określony rodzaj energii.

Energia jest intuicyjna w użyciu, gdyż spełnia prawo zachowania energii:

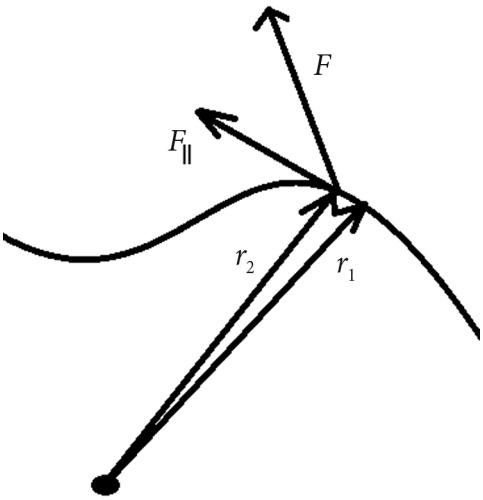
Zasada 6. Zasada zachowania energii
W układzie izolowanym od otoczenia energia jest zachowana.

Oznacza to, że jeżeli zsumujemy energie w układzie w danej chwili czasu i w innej, to będą one równe.

W dalszym ciągu jednak nie zostało podane, czym jest praca. Aby wykorzystać układ do wykonania pracy, potrzebujemy definicji samej pracy. Praca W jest zdefiniowana przez sumę[18] elementarnych przesunięć pomnożonych przez siłę działającą wzdłuż kierunku tego przesunięcia. Taka elementarna praca to

$$\Delta W = \left|\vec{F}_{\|}\right|\left|\Delta\vec{r}\right|, \qquad \text{(B.17)}$$

gdzie $\vec{F}_{\|}$ jest długością składowej siły wzdłuż przesunięcia $\Delta\vec{r}$ (ilustracja B.13).



Ilustracja B.13. Ciało porusza się po drodze, wzdłuż której działa siła $\vec{F}$. Rzut tej siły na kierunek styczny do krzywej to $\vec{F}_{\|}$. Wektor przesunięcia to różnica $\Delta\vec{r} = \vec{r}_2 - \vec{r}_1$

Praca może zależeć od drogi, wzdłuż której ją obliczamy, jednak w przypadku specjalnej klasy sił – sił potencjalnych – nie zależy. Takie siły, dla których praca jest różnicą energii na obu końcach drogi, nazywamy siłami potencjalnymi, czyli takimi siłami, dla których możemy zdefiniować potencjał. Wówczas nie musimy martwić się o drogę, a pracę na dowolnej drodze pomiędzy punktami A i B obliczamy jako różnicę energii w tych punktach:

$$W = E_B - E_A. \qquad \text{(B.18)}$$

Jest to bardzo użyteczne uproszczenie w obliczeniach. Okazuje się, że wiele sił występujących w zagadnieniach z robotyki jest potencjalnych – grawitacja, elek-

[18] Tutaj znowu pojawia się suma, którą rozpatrujemy w granicy przesunięć dążących do zera. Jak poprzednio, taką graniczną sumę możemy zdefiniować przez pojęcie całki. Zainteresowanych odsyłam do książki [20].

trostatyka. Dla przypomnienia, siła tarcia nie jest potencjalna, gdyż im dłuższą drogę przebędziemy, tym większą pracę ta siła wykona – praca zależy od drogi.

Zauważ, że wzór (B.18) ma bardzo ciekawą symetrię. Jeżeli zmienimy każdą z energii o pewną stałą C, tzn.

$$E_A \to E_A + C, \quad E_B \to E_B + C, \quad \text{(B.19)}$$

praca (B.18) się nie zmieni. Ta stała ustala poziom odniesienia dla energii i może być skończona lub nieskończona[19]. W praktyce użyteczna i mierzalna jest praca/różnica energii, a nie sama energia, która jest jedynie narzędziem do obliczania pracy.

B.3. Maszyny proste

Jako przykład zastosowania powyższych praw fizyki poniżej opisano maszyny proste, które można spotkać podczas konstrukcji robotów.

Maszyny proste to pasywne elementy, co oznacza, że nie posiadają wewnętrznych źródeł energii, zatem praca, którą wykonamy na maszynie prostej, jest tym, co możemy otrzymać[20].

[19] Problemy z nieskończoną wartością energii pojawiają się na fundamentalnym poziomie opisu świata – w kwantowej teorii pola. Jednak to nie jest trudność, bo prawdziwą wartość ma praca, a nie energia. Jeżeli jesteś zainteresowany poznaniem ciekawego świata kwantowej teorii pola i współczesnej fizyki teoretycznej, to możesz zajrzeć do publikacji [19].

[20] Tutaj pomijamy pracę straconą na pokonanie sił tarcia. Innymi słowy zakładamy, że sprawność maszyny, zdefiniowana jako praca otrzymana do pracy włożonej, jest dokładnie równa 1.

B.3.1. Dźwignia

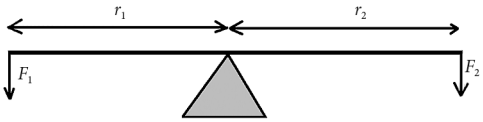
Pierwszą maszyną jest dźwignia, której schemat przedstawiono na ilustracji B.14. Dźwignia może obracać się wokół punktu podparcia. Zatem stosujemy tu prawa ruchu obrotowego. Załóżmy, że ilustracja B.14 przedstawia sytuację w równowadze. Wówczas moment siły generowany przez siłę $\vec{F}_1$ oraz analogiczny moment siły $\vec{F}_2$ równoważą się. Zatem mamy

$$F_1 r_1 = F_2 r_2, \quad \text{(B.20)}$$

czyli

$$\frac{F_2}{F_1} = \frac{r_1}{r_2}, \quad \text{(B.21)}$$

co jest wzorem na przekładnię dźwigni.



Ilustracja B.14. Dźwignia

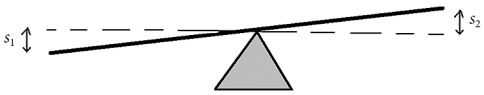
Jak możemy praktycznie wykorzystać ten wzór? Jeżeli ramiona są równe ($r_1 = r_2$), to aby utrzymać siłę $\vec{F}_2$, musimy przyłożyć po drugiej stronie siłę o takiej samej wartości. Zatem nic nie zyskujemy. Jednak jeżeli ramiona nie są równej długości, to równoważące się siły nie muszą mieć takiej samej wartości. W przypadku gdy chcemy podnieść coś ciężkiego w polu grawitacyjnym, część dźwigni, do której przykładamy siłę, powinna być bardzo długa, abyśmy nie musieli używać dużej siły. To jest niemałe ułatwienie.

Spróbujmy jeszcze obliczyć zysk na sile, wiedząc, że dźwignia jest urządzeniem pasywnym. Wyobraźmy sobie, że układ sił z poprzedniego rysunku obrócił dźwignię do pozycji pokazanej na ilustracji B.15. Prace wykonane przez obie siły są równe, zatem

$$W = F_1 s_1 = F_2 s_2, \tag{B.22}$$

czyli

$$\frac{F_2}{F_1} = \frac{s_1}{s_2}. \tag{B.23}$$



Ilustracja B.15. Dźwignia po małym przesunięciu

Wykorzystując podobieństwo trójkątów utworzonych przez dźwignię przed i po przesunięciu, dochodzimy do

$$\frac{r_1}{r_2} = \frac{s_1}{s_2}, \tag{B.24}$$

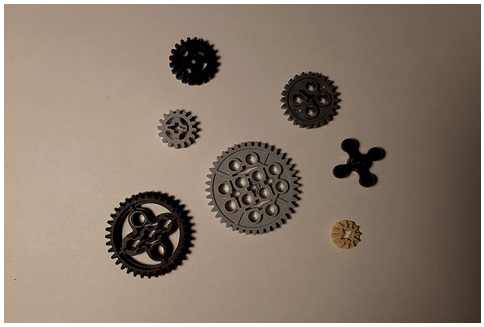
czyli odtwarzamy wzór (B.21).

B.3.2. Przekładnia

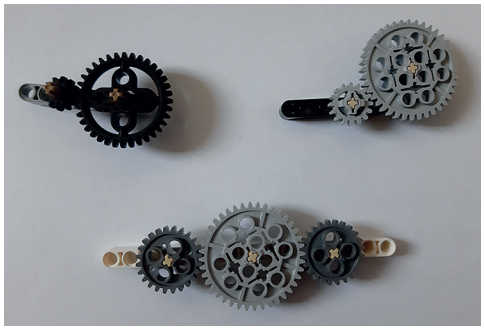
Przekładnie służą do przełożenia ruchu z elementu napędowego do elementu napędzanego. Ze względu na sposób transferu ruchu przekładnie dzielimy na kilka typów. Dla nas najważniejsze będą przekładnie zębate, w których używamy kół zębatych przedstawionych na ilustracji B.16. Koła te możemy łączyć w przekładnie, jak to pokazuje ilustracja B.17. Jest oczywiste, że aby przekładnia działała, każde koło zębate musi mieć zęby rozłożone na obwodzie z tą samą gęstością, a wielkości zębów muszą być dopasowane.

Przekładnie możemy podzielić ze względu na stosunek prędkości elementów napędowego i napędzającego na:

- reduktory lub przekładnie redukujące – zmniejszają prędkość obrotów;
- multiplikatory lub przekładnie multiplikujące – zwiększają prędkość obrotów.



Ilustracja B.16. Przykłady kół zębatych



Ilustracja B.17. Przykłady przekładni zębatych

Policz, jaki jest stosunek prędkości elementu napędzającego i napędowego. Wykorzystaj fakt, że w punkcie styku prędkości liniowe dla elementu napędzającego v_1 i napędzanego v_2 muszą być takie same. Jeżeli promienie tych elemen-

tów to odpowiednio r_1 i r_2, a prędkości kątowe ω_1 oraz ω_2, wówczas:

$$v_1 = r_1\omega_1 = r_2\omega_2 = v_2. \qquad \text{(B.25)}$$

Zatem stosunek prędkości kątowych tych elementów zwany przełożeniem przekładni to

$$\frac{\omega_1}{\omega_2} = \frac{r_2}{r_1}. \qquad \text{(B.26)}$$

Aby obliczyć liczbę zębów, należy wykorzystać fakt, że gęstość liniowa zębów ρ na obu kołach zębatych jest taka sama. Zatem na pierwszym kole mamy $N_1 = 2\pi r_1 \rho$ zębów, a na drugim $N_2 = 2\pi r_2 \rho$. Ich stosunek to

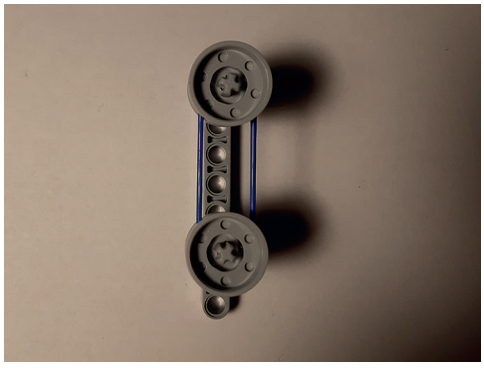
$$\frac{r_2}{r_1} = \frac{N_2}{N_1}, \qquad \text{(B.27)}$$

a zatem stosunek prędkości kątowych to

$$\frac{\omega_1}{\omega_2} = \frac{r_2}{r_1} = \frac{N_2}{N_1}. \qquad \text{(B.28)}$$

Przełożenie przekładni wyraża się przez stosunek liczby zębów. Więc licząc jedynie liczbę zębów na kołach zębatych, możemy określić, jak szybko będzie poruszało się jedno koło w stosunku do drugiego.

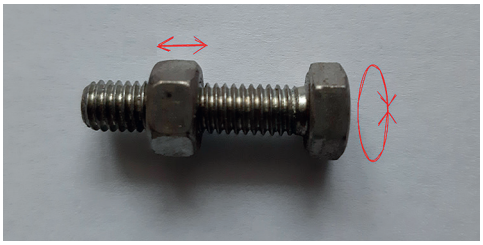
Innym przykładem są przekładnie cięgnowe, w których przełożenie napędu odbywa się przy pomocy cięgna, którym może być łańcuch (jak w rowerze) lub pas, jak to przedstawiono na ilustracji B.18. Dla tych przekładni również prawdziwy jest wzór na przełożenie przekładni (B.26).



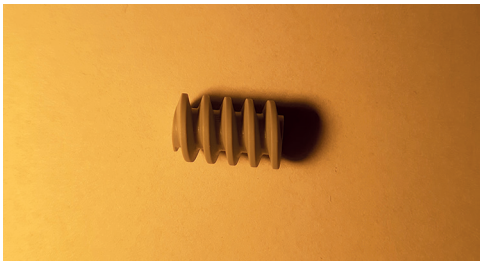
Ilustracja B.18. Przykład przekładni pasowej. Jako cięgna użyto gumki z zestawu rozszerzonego

B.3.3. Śruba

Kolejnym użytecznym elementem będzie śruba. Jest ona wykorzystywana przy zamianie ruchu obrotowego na posuwisty. Przykład śruby przedstawiono na ilustracjach B.19 i B.20. Odległość pomiędzy dwiema sąsiednimi „wypustkami" nazywa się skokiem gwintu l i mówi ona, o ile przesunie się element



Ilustracja B.19. Typowa śruba. Ruch obrotowy jest zamieniany na ruch postępowy



Ilustracja B.20. Śruba dostępna w zestawie

podążający wzdłuż gwintu po jednym obrocie. Zatem jeżeli śruba obróci się o kąt ϕ, element poruszający się wzdłuż gwintu przesunie się o

$$h = l\frac{\phi}{2\pi}. \qquad \text{(B.29)}$$

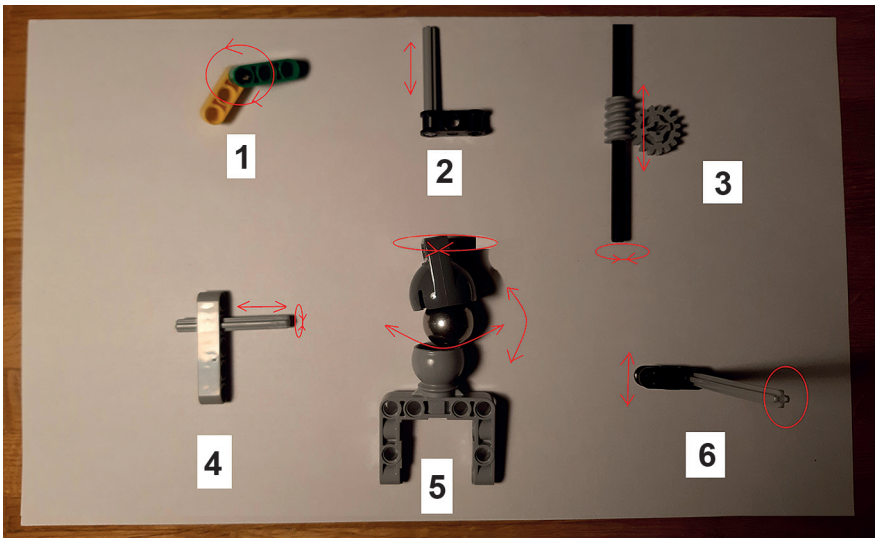
Spróbujmy teraz przeanalizować śrubę pod kątem pracy, jaką wykonuje. Jeden obrót siłą F_1 przyłożoną stycznie do brzegu śruby o promieniu r wymaga pracy $W_1 = 2\pi r F_1$. Generuje to przesunięcie elementu podążającego wzdłuż gwintu o l, a siła, która ten element popycha, to F_2, więc praca to $W_2 = lF_2$. Ponieważ śruba jest również elementem pasywnym, obie prace muszą być takie same, a zatem

$$\frac{F_2}{F_1} = \frac{2\pi r}{l}. \qquad \text{(B.30)}$$

Jest to, analogicznie jak w przypadku dźwigni, wzór na przekładnię śruby.

B.4. Ogólne spojrzenie na konstrukcję układów mechanicznych

W nowszym podejściu do konstrukcji maszyn zapoczątkowanym przez Franza Reuleaux [21] istotny nacisk kładzie się na analizę par kinematycznych – ruchomych elementów łączących stałe części mechanizmu. Zatem cały mechanizm



Ilustracja B.21. Niższe pary kinematyczne z najmniejszą liczbą stopni swobody. Kolorem czerwonym zaznaczono możliwości ruchu – stopnie swobody. 1) przegub obrotowy o jednym stopniu swobody – obrót; 2) przegub pryzmatyczny o jednym stopniu swobody – ruch; 3) przegub śrubowy o dwóch stopniach swobody – ruch obrotowy śruby i ruch postępowy (w tym przypadku przełożony na ruch obrotowy trybu); 4) przegub cylindryczny – dwa stopnie swobody – obrót i ruch; 5) przegub sferyczny z trzema stopniami swobody – ruch po sferze z dwoma stopniami i wzajemny obrót; 6) przegub typu płaska para – ruch po płaszczyźnie z dwoma stopniami swobody i obrót

możemy rozłożyć na zestaw brył sztywnych połączonych różnego typu parami kinematycznymi. Połączenia te możemy sklasyfikować ze względu na to, ile niezależnych ruchów, zwanych stopniami swobody przegubu, dopuszczają. Te najprostsze, o najmniejszej liczbie stopni swobody, przedstawione są na ilustracji B.21.

Techniki umożliwiające konstrukcję mechanizmu o odpowiednim zakresie ruchów oraz odpowiednim przełożeniu sił nazywamy syntezą kinematyczną [21].

Dodatek C

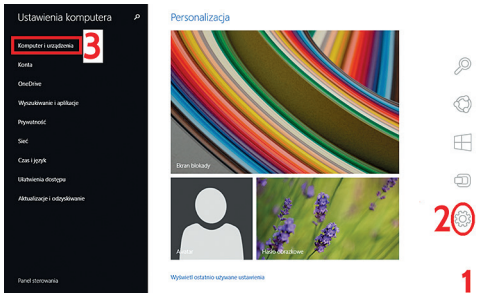
Instalacja oprogramowania

Kontroler jest minikomputerem opartym na systemie Linux. Posiada on moduł Bluetooth, WiFi, a także złącze miniUSB do połączenia z komputerem lub smartfonem/tabletem, z których możemy ten klocek programować. W kolejnych sekcjach opisano szczegóły instalacji i konfiguracji.

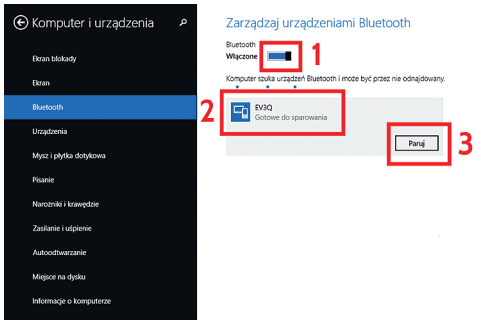
C.1. PC z systemem Windows, połączenie przez Bluetooth – zalecane

Dla systemów operacyjnych Windows 7, 8, 10 istnieje wygodne środowisko programistyczne LEGO Mindstorms EV3 Software [2]. Ten sposób programowania kontrolera LEGO® Mindstorms® EV3 jest zalecany. Oprogramowanie należy pobrać i zainstalować w standardowy sposób.

Zalecanym sposobem komunikacji pomiędzy klockiem-kontrolerem i komputerem klasy PC z systemem Windows jest użycie połączenia Bluetooth. Sposób połączenia przedstawiono na przykładzie systemu Windows 8. Pierwszym krokiem jest sparowanie klocka z komputerem. Na początku musisz włączyć (na klocu) połączenie Bluetooth i ustawić jego widoczność. Możesz też klockowi nadać unikalną nazwę, dzięki której rozpoznasz go podczas parowania. W tym przypadku będzie to nazwa EV3Q. Włączenie interfejsu Bluetooth przedstawia ilustracja C.1. Następnie aktywuj połączenie Bluetooth, wyszukaj nazwę klocka na liście (pamiętaj, aby na kontrolerze włączyć Bluetooth i jego widoczność) i sparuj urządzenia, jak to przedstawia ilustracja C.2. Po naciśnięciu przycisku *Paruj*, na ekranie kontrolera pojawi się kod. Zaakceptuj go, a następnie przepisz do okna na komputerze.

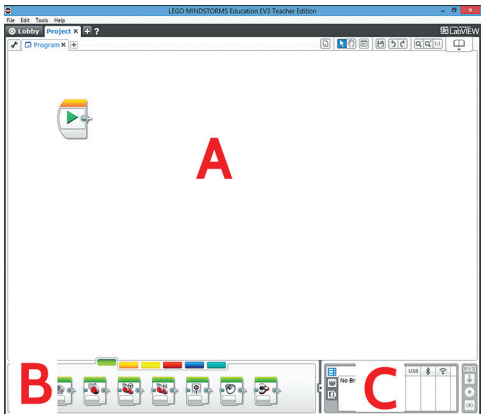


Ilustracja C.1. Przesuń myszkę w róg ekranu – 1, następnie wybierz ikonę ustawień – 2, a później wybierz *Komputer i ustawienia* – 3



Ilustracja C.2. Włącz interfejs Bluetooth – 1, odczekaj, aż na liście pojawi się nazwa klocka – 2, następnie sparuj urządzenia – 3

Jeżeli wszystko przebiegło poprawnie, po uruchomieniu programu LEGO i rozpoczęciu tworzenia nowego programu twój klocek powinien być widoczny w miejscu C na ilustracji C.3.



Ilustracja C.3. Okno nowego programu. Sparowany klocek powinien być widoczny w sekcji C

C.2. PC z systemem Windows, połączenie przez kabel USB

Oprogramowanie LEGO Mindstorms EV3 Software [2] po uruchomieniu i połączeniu kablem USB klocka do komputera powinno wykryć klocek. Od tego momentu jest możliwe programowanie w środowisku programistycznym i wysyłanie programu do klocka.

Połączenie przy użyciu kabla USB powinno działać na każdym sprzęcie. Ma ono jednak kilka wad:

- Należy pamiętać, żeby nie uruchamiać programu w momencie, gdy kabel jest podłączony. Jeżeli programujesz urządzenie, które będzie się poruszać, i uruchomisz program podczas połączenia kablem, może to prowadzić w najlepszym wypadku do odpięcia kabla, a w najgorszym do uszkodzenia portów USB.
- Port USB na klocku jest solidnie osadzony, jednak przy częstym używaniu lub użyciu zbyt dużej siły może on zostać poluzowany lub uszkodzony, jak w każdym urządzeniu elektronicznym.
- Połączenie przy pomocy kabla jest mało elastyczne, aczkolwiek najbardziej niezawodne i najprostsze do wykonania.

C.3. PC z systemem Linux

Niestety, nie istnieje środowisko programistyczne dla systemu Linux, które jest analogiczne do tego z systemu Windows. Jednak możesz użyć np. systemu Windows zainstalowanego na wirtualnej maszynie dla systemu Linux.

Poniżej opisano konfigurację dla maszyny wirtualnej Oracle Virtualbox dla systemu Ubuntu. Procedura instalacji na systemie Linux jest następująca:

- Zainstaluj Oracle Virutalbox [23] przez systemowy manager pakietów.
- Stwórz maszynę wirtualną i zainstaluj na niej legalny system Windows w wersji 7, 8 lub 10.
- Zainstaluj dodatek gościa (*guest addition*) Windowsa.
- Zainstaluj rozszerzenie *extension pack* poleceniem wpisanym w terminalu: sudo apt install virtualbox--ext-pack.
- W celu zapewnienia dostępu wirtualnej maszynie do portów USB dodaj siebie do grupy *vboxuseres* poleceniem wpisanym w terminalu: sudo adduser $USER vboxusers.
- Wyloguj się i ponownie zaloguj, aby zmiany weszły w życie.

Od tego momentu możesz przekazywać porty USB bezpośrednio do Virtualbox, klikając na ikonkę USB w prawym dolnym rogu uruchomionej maszyny.

C.4. Smartfon/tablet

Z poziomu systemu operacyjnego Android możemy połączyć się z klockiem przy pomocy modułu Bluetooth. Na systemie Android należy zainstalować program Lego® Mindstorms® Commander [24]. Komentarze do aplikacji sugerują, że użytkownicy mają notoryczne problemy z nawiązaniem komunikacji z klockiem, co również zdarzyło się autorowi skryptu. Komentarze również sugerują, że nie wszystkie opcje programistyczne są udostępnione w mobilnej aplikacji. Jednak ze względu na kompletność prezentacji opisano w skrypcie teoretyczny proces nawiązania połączenia.

W celu połączenia na klocku należy włączyć Bluetooth i ustawić widoczność, a jednocześnie wyłączyć opcję połączenia dla iPhone'a/iPada. Wówczas z poziomu systemu Android należy włączyć aplikację Commander i wyszukać urządzenie EV3. Na forach internetowych pojawiają się też sugestie, że jeżeli proces wyszukiwania urządzenia nie daje rezultatu, należy sparować urządzenia, a następnie powtórzyć wyszukiwanie z poziomu aplikacji Commander.

Dla iPhone'a/iPada istnieje również podobna aplikacja. Tym razem na klocku należy dodatkowo ustawić opcję połączenia z tymi urządzeniami.

Dodatek D

Praca w grupie

W tej części zostanie opisana efektywna praca w grupie zespołu programistycznego.

D.1. Inżynieria oprogramowania

W rozwoju programisty wyróżniamy kilka ważnych etapów:

1. Nauka programowania w wybranym języku programowania.
2. Algorytmy i struktury danych – nauka wykorzystania cegiełek (algorytmów i struktur danych), z których można złożyć program i w wybranym języku można je zaimplementować [25].
3. Inżynieria oprogramowania – nauka tworzenia złożonych programów i nauka pracy w grupie [26].

Ostatni z tych elementów traktuje o tworzeniu skomplikowanych projektów informatycznych, skutecznej pracy w grupie oraz technikach zarządzania grupą tak, aby projekt miał szansę ukończenia w wyznaczonym czasie.

Jest wiele efektywnych technik pracy w grupie. Tutaj skupiono się jednak na metodzie Scrum, która jest jedną z tzw. metodyk zwinnych (ang. *Agile*).

D.2. Scrum

Metodyka Scrum została dokładnie opisana w bezpłatnym poradniku *The Scrum Guide* [27]. Poniżej przedstawiono jej uproszczoną wersję.

W grupie pracującej zgodnie ze Scrumem mamy do czynienia z następującymi podmiotami:

- właściciel produktu – osoba zlecająca wykonanie programu lub jej reprezentant;
- zespół deweloperski – zespół pracujący nad projektem, zazwyczaj składa się z 3 do 9 osób;
- Scrum Master – osoba dbająca o efektywność zespołu oraz pracę z metodyką Scrum.

Proces produkcji zaczynamy od rozmowy z właścicielem produktu. W rozmowie tej należy przedyskutować wszystkie elementy, które są przez niego wymagane. Wynikiem końcowym jest zbiór przypadków użycia opisujących interakcję z projektowanym systemem.

Następnym krokiem jest opracowanie rejestru wymagań (ang. *Product Backlog*) zawierającego wszystkie elementy wymagane przez właściciela produktu. Są one najczęściej wymienione w punktach. Te punkty zostają następnie przepisane na szczegółowe zadania dla zespołu. Zadania te są zapisywane na kartkach (np. samoprzylepnych). Podczas tych prac należy również oszacować czas wykonania poszczególnych zadań i całości oraz ryzyko niepowodzenia. Następnie przygotowuje się tablicę scrumową, która jest podzielna na trzy części: „Do zrobienia" (*To Do*), „W trakcie" (*In Progress*), „Zakończone" (*Done*). Karteczki przykleja się do części „Do zrobienia".

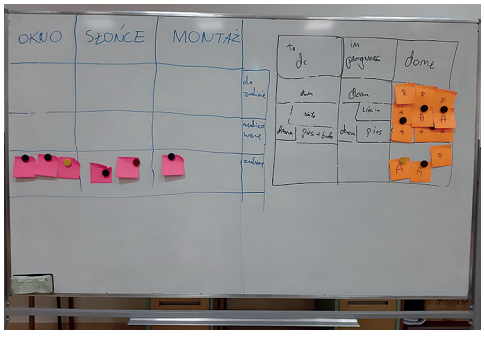
Po tym rozpoczyna się praca nad projektem w cyklach nazywanych sprintami. Pojedynczy sprint nie powinien trwać dłużej niż miesiąc. Na początku każdego dnia zespół spotyka się na krótkim (maksymalnie 15-minutowym) spotkaniu, na którym każdy członek zespołu pobiera zadanie przez zapisanie swojego identyfikatora na karteczce i przesunięciu jej do części „W trakcie". Karteczki z zadaniami zakończonymi przesuwane są do części „Zakończone". Każdy członek zespołu szacuje czas potrzebny na ukończenie rozpoczętego zadania oraz informuje wszystkich o postępach i ewentualnych problemach z ukończeniem. Spotkaniem zarządza Scrum Master. Po zakończeniu sprintu następuje spotkanie (*sprint review*), na którym prezentowany jest produkt i ustalany jest termin spotkania z właścicielem produktu, aby zebrać wymagania do następnej iteracji. Po każdej iteracji zespół organizuje spotkanie (*retrospective*), na którym w swobodnej atmosferze omawiane są doświadczenia ze sprintu i planowane są ulepszenia w pracy zespołu. W każdej iteracji powstaje coraz bardziej dopracowany produkt.

D.3. Symulacja

W tej sekcji przedstawimy symulację pracy grupy nad projektem przy użyciu metody Scrum. Biorą w niej udział dwie lub więcej grup. Grupy wyznaczają sobie (jako właściciele produktu) zadania. Może to być wykonanie konstrukcji z plasteliny lub z omawianych w tej książce klocków.

Dzień w takiej symulacji może trwać 10–20 minut. Ważne jest to, aby zadbać o spotkania przy tablicy scrumowej (ilustracja D.1) na początku każdego dnia. Spotkania te koordynuje wyznaczony w grupie Scrum Master.

Po zakończonym sprincie zespół spotyka się z właścicielem produktu i zdaje sprawozdanie oraz pokazuje zamówiony



Ilustracja D.1. Przykład wykonania Tablicy scrum dla dwóch grup przy użyciu jednej tablicy i kartek samoprzylepnych

produkt, a także planuje kolejny sprint zgodnie z przyjętą metodą, jeżeli jest czas na taką iterację.

Należy zadbać o to, aby taka symulacja była w miarę realistyczna. Powinno się zwrócić uwagę na to, aby podczas rozmowy z właścicielem produktu delegacja zespołu deweloperów dokładnie wypytała właściciela o wymagania. W rzeczywistym świecie zaniedbania na początku mogą skutkować ostatecznym produktem niezgodnym z wymaganiami, a to może prowadzić do braku zapłaty za wykonaną pracę lub pozwu sądowego o odszkodowanie. Należy uczulić osoby biorące udział w takim spotkani, że każde wymaganie właściciela produktu należy spisać i potwierdzić, aby mieć dowód zlecenia w przypadkach przyszłych spornych kwestii.

Kolejnym ważnym elementem, którego nie powinno się zaniedbywać, jest planowanie sprintu. Jest to etap istotny i należy mu poświęcić dużo czasu. Źle zaplanowany sprint lub źle opracowany rejestr wymagań, a przez to nieprawidłowo zlecone zadania dla zespołu, mogą skutkować źle wykonanym projektem lub jego niewykonaniem. Może to prowadzić również do sporów z właścicielem produktu/zleceniodawcą.

Pytania kontrolne

Poniżej znajduje się lista przykładowych pytań do uczestników szkolenia. Test należy zrobić na początku zajęć (test referencyjny), a także na końcu. Porównując wyniki, otrzymuje się obraz rzeczywistej wiedzy, którą nabyli uczniowie korzystający z tego skryptu.

1. O czym traktuje pierwsze prawo Newtona?
   (a) Zasada akcji i reakcji.
   (b) Przyśpieszenie jest proporcjonalne do siły i odwrotnie proporcjonalne do masy: $F = ma$.
   (c) Ruch swobodny w układzie inercjalnym to spoczynek lub ruch jednostajny po linii prostej.
2. O czym traktuje drugie prawo Newtona?
   (a) Zasada akcji i reakcji.
   (b) Przyśpieszenie jest proporcjonalne do siły i odwrotnie proporcjonalne do masy: $F = ma$.
   (c) Ruch swobodny w układzie inercjalnym to spoczynek lub ruch jednostajny po linii prostej.
3. O czym traktuje trzecie prawo Newtona?
   (a) Zasada akcji i reakcji.
   (b) Przyśpieszenie jest proporcjonalne do siły i odwrotnie proporcjonalne do masy: $F = ma$.
   (c) Ruch swobodny w układzie inercjalnym to spoczynek lub ruch jednostajny po linii prostej.
4. Czy znasz przykłady maszyn prostych? Wymień przynajmniej dwie.
5. Maszyny proste są elementami…
   (a) aktywnymi.
   (b) pasywnymi.
6. Do czego służy mechanizm różnicowy?
   (a) Do obliczania różnicy dwóch liczb.
   (b) Do nadawania przyśpieszenia robotom.
   (c) Do umożliwienia skrętu samochodom przy jednoczesnym napędzie kół.
7. Co mierzymy w hercach?
   (a) Częstotliwość fali.
   (b) Energię układu.
   (c) Moc maszyny.

8. W jakim celu zostało wymyślone pojęcie pojazdów Breitenberga?
   (a) Aby nazwać pojazdy ścigające się.
   (b) Aby nazwać pojazdy do zawodów „sumo robotów".
   (c) Aby nazwać pojazdy reagujące na światło.
9. Co to jest test Turinga?
   (a) Test pozwalający określić, czy robot potrafi manipulować przedmiotami podobnie jak człowiek.
   (b) Test pozwalający określić, czy sztuczna inteligencja jest odróżnialna podczas rozmowy od człowieka.
   (c) Test pozwalający określić, czy robot jest podobny do maszyny Turinga.
10. Do czego służy żyroskop?
    (a) Do pomiaru orientacji.
    (b) Do pomiaru przyśpieszenia.
    (c) Do pomiaru pracy wykonanej przez maszynę.

Odpowiedzi
1c, 2b, 3a, 4. dźwigniа, śrubа, 5b, 6c, 7a, 8c, 9b, 10a

Bibliografia

[1] Strona Lego® Mindstorms® EV3, https: //education.lego.com/en-us/support/ mindstorms-ev3/getting-started (data dostępu: 11.12.2020).

[2] Oprogramowanie dla Lego® Mindstorms® EV3, https://education.lego.com/ en-us/ downloads/mindstorms-ev3/software (data dostępu: 11.12.2020).

[3] Opis użycia języka Python w Lego® Mindstorms® EV3, https://education.lego. com/en-us/support/ mindstorms-ev3/python-for-ev3 (data dostępu: 11.12.2020).

[4] Jason R. Briggs, *Python dla dzieci. Programowanie na wesoło*, Wydawnictwo Naukowe PWN, 2015.

[5] System Linux dla Lego® Mindstorms® EV3, https://www.ev3dev.org/ (data dostępu: 11.12.2020).

[6] MEMS – układ SparkFun MPU-9250 9DoF, https://www.sparkfun.com (data dostępu: 11.12.2020).

[7] Pixabay, https://www.pexels.com (data dostępu: 11.12.2020).

[8] Daniele Benedettelli, *Laboratorium LEGO Mindstorms EV3. Buduj, programuj i eksperymentuj*, Helion, 2014.

[9] Henryk Brandys, *LEGO Mindstorms EV3. Podstawy programowania*, Helion, 2020.

[10] Yoshihito Isogawa, *Lego Mindstorms EV3. Niesamowite projekty*, Helion, 2015.

[11] Megan H. Rothrock, *LEGO. Księga przygód*, Helion, 2017.

[12] ShanghAI Lectures, https://shanghai-lectures.github.io/ (data dostępu: 11.12.2020).

[13] Greg S. Hornby, Al Globus, Derek S. Linden, Jason D. Lohn, *Automated antenna design with evolutionary algorithms*, American Institute of Aeronautics and Astronautics, 2006.

[14] *Pracownik przyszłości*, raport, https: //www.samsung.com/pl/campaign/ brand/ pracownik-przyszlosci/ (data dostępu: 11.12.2020).

[15] David Halliday, Robert Resnick, Jearl Walker, *Podstawy fizyki*, tomy 1–5, PWN, 2019.

[16] Andrzej Januszajtis, *Fizyka dla Politechnik*, tomy 1–3, PWN, 1977.

[17] Jan Kurzyk, *Jak działa GPS*, Fizyka w Szkole 1, 2017, http://www.aspress.com. pl/gps/ (data dostępu: 11.12.2020).

[18] Neil Ashby, *Relativity in the Global Positioning System*, Living Rev Relativ, 2003, 6(1), https//dx.doi.org/10.12942%2Flrr-2003-1 (data dostępu: 11.12.2020).

[19] Wouter Schmitz, *Particles, Fields and Forces: A Conceptual Guide to Quantum Field Theory and the Standard Model*, Springer, 2019.
[20] Jason Wilkes, *Spal pracownię matematyczną... i sam sobie wymyśl matematykę*, Prószyński i S-ka, 2017.
[21] John J. Uicker, Jr., Gordon R. Pennock, Joseph E. Shigley, *Theory of Machines and Mechanisms*, Oxford University Press, 2017.
[22] https://www.csc.fi/web/elmer (data dostępu: 11.12.2020).
[23] Oprogramowanie Oracle VirtualBox, https://www.virtualbox.org/ (data dostępu: 11.12.2020).
[24] Oprogramowanie Lego® Mindstorms® EV3 dla systemu Android, https://play.google.com/store/apps/details?id=com.lego.mindstorms.robotcommander (data dostępu: 11.12.2020).
[25] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Stein Clifford, *Wprowadzenie do algorytmów*, PWN, 2017.
[26] Perdita Stevens, *UML. Inżynieria oprogramowania*, Helion, 2007.
[27] Jeff Sutherland, Ken Schwaber, *The Scrum Guide*, https://www.scrumguides.org/ (data dostępu: 11.12.2020).