ZESPÓŁ SZKÓŁ ELEKTRYCZNYCH

IM. PROF. JANUSZA GROSZKOWSKIEGO

W BIAŁYMSTOKU

Poradnik programowania robota

Lego Mindstorms

Autorzy:

1.Czaplicki Krystian
2.Kowalczuk Krystian

Pod kierunkiem:

mgr inż. Raciborski Jarosław

Białystok 2010

Przedmowa

Czy lubicie programować?

Oczywiście, że lubimy.

Programowanie jest strasznie fajne piszesz polecenie wypisz (zazwyczaj po angielsku albo w bardziej przyjaznym - dla naszego nauczyciela języku) potem podajesz, co ma wypisać komputer musisz pamiętać, w jakie znaki ująć i efekt- widzimy piękny napis (najlepiej bez polskich znaków) na czarnym tle magicznego DOSa o treści WITAJ SWIECIE. Prawda, że proste i fascynujące. Tylko, komu to potrzebne. Nikomu, ale zawsze od tego zaczynamy. No i fajnie. Ale potem przechodzimy do takich spraw, które są bardziej przydatne, a wiec typy, deklaracje zmiennych, warunki, pętle, tablice, struktury itd. Na pewnym etapie po rozwiązaniu 150 zadań zaczynających sie poleceniem "Napisz program..." Dochodzimy do wniosku - po co ja to robię. Owszem, aby dostać kolejną szóstkę w tym tygodniu z programowania - to niewątpliwie dobry powód. Po drugie mam satysfakcję, że pokonałem maszynę krzycząc z wrażenia "Wreszcie sie skompilował". Po trzecie jestem wspaniałym programistą i mogę pochwalić się przed kolegami programistami, którym sie to nie udaje tak często jak mi. Ale gdy spotkam się z kolegami, którzy nie są informatykami i mówię im o pętli for, to widzę na początku fascynację moją osobą, ale potem znudzenie, bo tak naprawdę brak efektów wizualnych mojego poniekąd sukcesu. O!!!- mówi kolega -ja wczoraj o 2 w nocy pokonałem "ROBOTIXA" mając w rękach "BAZUKI" (przepraszam, że przekręcam nazwy, ale w gry nie gram) i to jest dopiero sukces, a swoją pętlę FOR czy jak tam ją nazwałeś zostaw dla swego nauczyciela.

Pomyśleliśmy, więc z kolegą jak nauczyć, zainteresować programowaniem tych, których to nie bawi, aby zamiast każdą wolną chwilę przeznaczyć na zabijanie jakichś magów czy innych krasnali lub olbrzymów zainteresowali sie programowaniem. Wpadliśmy na pomysł, aby program coś fizycznie wykonywał, był widzialny nie tylko na ekranie monitora, ale można było zastosować go w praktyce. Oczywiście napisać grę. To dobry pomysł i na pewno by chwycił. Ale dalej tylko ekran i klawiatura. "Biegając" po Internecie natknęliśmy się na coś, co nie jest już nowością, bo już jesteśmy DOROŚLI mianowicie klocki LEGO. Ale nie są to zwykłe klocki - bo można je programować. Mało tego mają czujniki różnego typu, kamerkę i wiele innych ciekawych rzeczy. Można je prosto programować pisząc a w zasadzie układając proste algorytmy blokowe, ale też można użyć profesjonalnego języka programowania, aby te klocki poruszyć. My lubimy programować, ale czy wszyscy lubią w naszej klasie - na pewno NIE. My lubimy układać klocki, (mimo że mamy już 17 lat), ale czy wszyscy będą to lubić w naszej klasie - na pewno TAK.

Dlatego celem naszej pracy jest stworzenie na początek kilku zestawu ćwiczeń dla początkujących programistów, aby można było na ich podstawie złożyć robota z klocków lego, a następnie go zaprogramować. Mamy jeszcze dalsze plany: stworzyć koło programowania robotów w naszej szkole i sami to koło prowadzić. Życzymy wszystkim i sobie MIŁEJ ZABAWY.

Poradnik ten stanowić ma pomoc dydaktyczną na zajęciach programowania od podstaw układania klocków, po tworzenie i wgrywanie oprogramowania i uruchamiania robota.

1. Wprowadzenie

Poradnik ma wprowadzać czytelnika w świat przystępnego programowania i podnosić kwalifikacje z zakresu:

- projektowania rozwiązań algorytmicznych,

- kodowania algorytmów za pomocą Lego Mindstorms Software i RobotC

- testowania algorytmów na robocie

- pracy w drużynie.

W poradniku zamieszczono:

- wykaz umiejętności, jakie powinien posiadać czytelnik by bez problemów poradzić sobie z programowaniem robota,

- cele kształcenia, czyli umiejętności jakie posiądziesz po przeprowadzeniu kursu,

- materiał nauczania, czyli wiadomości, których nauczysz się na kursie,

- zestaw pytań i ćwiczeń w celu sprawdzenia stopnia opanowania przez czytelnika materiału,

- literatura uzupełniająca.

W razie wątpliwości czy trudności z wykonaniem zadania należy skonsultować się z nauczycielem prowadzącym.

2. Wymagania wstępne

Przystępując do realizacji zadań z programowania jak i składania robota czytelnik powinien umieć:

- podstawy języka C

- umiejętność składania klocków lego i prostych układów elektrycznych

- umiejętność konfiguracji połączenia Bluetooth na jednostce centralnej

- podstawy obsługi komputera.

3. Cele kształcenia

Po zakończonym procesie kształcenia czytelnik będzie potrafił:

- określić pojęcia: algorytm, program, język programowania, serwomotor, sensor, sensor dotyku, sensor koloru, dalmierz ultradźwiękowy, kostka NXT.

- zbudować robota,

- podłączać dodatkowe sensory do robota,

- planować rozwiązania algorytmiczne do przedstawionych problemów,

- sprawdzać poprawność stosowanych algorytmów,

- posługiwać się środowiskami programistycznymi Lego Mindstorms Software, RobotC,

- rozwiązywać proste zadania programistyczne przy pomocy wyżej wymienionych środowisk programistycznych,

- programować w języku RobotC.

4. Materiał nauczania

4.1. Algorytm

Algorytm to przepis, który podaje wszelkie czynności, jakie należy wykonać, by osiągnąć rozwiązanie określonego problemu w skończonej liczbie kroków. W celu stworzenia algorytmu należy:

- rozważyć problem,
- wymyślić uszeregowany ciąg kroków rozwiązujący problem i zapisać go w odpowiedniej formie,
- przeanalizować algorytm w celu jego udoskonalenia i wyeliminowania wad,
- przenieść algorytm do wybranego przez siebie języka programowania.

Formy przedstawienia algorytmu:

- opis słowny,

- opis za pomocą listy kroków,

- opis w postaci schematu blokowego,

- opis za pomocą drzewa,

Pseudokod to uproszczona wersja typowego języka programowania. Symbole geometryczne używane w schemacie blokowym zostały zastąpione zdaniami w ojczystym języku, które opisują kodowany problem.

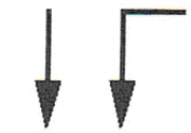
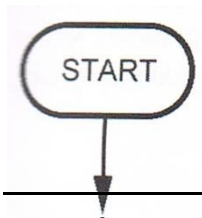
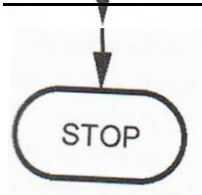
Drzewem nazywamy graficznym algorytm przedstawiony w postaci linii lub wektorów symbolizujących drogę wzdłuż której wykonywane są operacje arytmetyczno logiczne. Drogi mają wspólny początek, lecz inne wierzchołki końcowe.

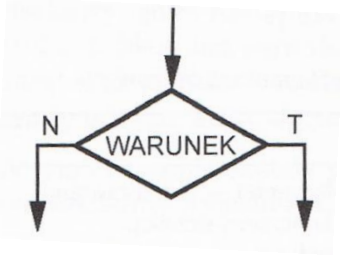
Lista kroków stanowi uporządkowany opis wszelkich czynności, jakie mają być wykonywane w ramach algorytmu. Krok to pojedyncza czynność realizowana w algorytmie. Ponieważ kolejność wykonywania ma znaczenie, poszczególne kroki w algorytmie są numerowane.

Opis słowny polega na podaniu czynności, które należy podjąć, by uzyskać oczekiwany efekt. Opis musi być zrozumiały dla odbiorcy.

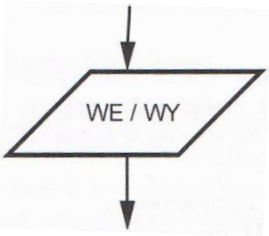
Schemat blokowy to plan algorytmu przedstawiony w formie graficznej struktury elementów zwanych blokami. Każdy blok zawiera informację o operacji, która ma być w nim wykonana. Pomiędzy blokami znajdują się połączenia, inaczej linie przepływu, określające kolejność wykonywania klatek schematu.

Tab. 1. Podstawowe elementy graficzne schematu blokowego

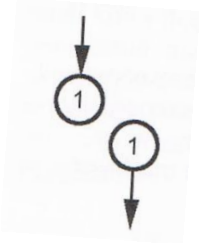
	Linia przepływu – przedstawiana jest w postaci linii zakończonej strzałką, która wskazuje kierunek przemieszczania się po schemacie.
	Blok początku – wskazuje miejsce rozpoczęcia algorytmu. Posiada jedno wyjście, a nie posiada wejścia. Na schemacie występuje tylko jedna klatka startowa.
	Blok końcowy – wskazuje miejsce zakończenia algorytmu. Posiada przynajmniej jedno wejście, a nie posiada wyjścia, może występować kilka takich bloczków w algorytmie.
	Blok wykonawczy – zawiera polecenie, które należy wykonywać wykorzystując dostarczone dane wejściowe. Uzyskany wynik jest wyprowadzany na zewnątrz klatki. Instrukcją może być podstawienie, operacja arytmetyczna, wprowadzanie danej.



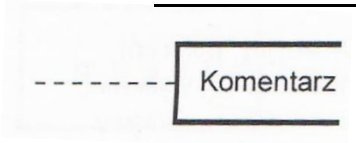
Blok warunkowy – sprawdza umieszczony w nim warunek i dokonuje wyboru tylko jednej drogi wyjściowej. Z bloku wychodzą dwa połączenia. Najczęściej oznaczone są T – tak, co oznacza, że warunek został spełniony; N – nie, warunek nie został spełniony.



Blok wejścia-wyjścia – poprzez ten blok wprowadzane są dane i wyprowadzane są wyniki. Blok posiada jedno połączenie wejściowe i jedno wyjściowe.



Bloki łącznikowe – stosowane są wtedy, gdy zachodzi konieczność kontynuowania rysunku w innym miejscu kartki lub na kolejnej stronie.



Blok komentarza – umieszcza się w nim opisy niezbędne do zrozumienia kolejnych poleceń w algorytmie. Treść komentarza nie ma żadnego wpływu na wynik

4.2.Program

Programem nazywamy skompilowaną do formy bitowej sekwencje symboli opisująca obliczenia zgodnie z pewnymi regułami zwanymi językiem programowania. Programy mogą być wykonywalne przez wszystkie urządzenia elektroniczne posiadające procesor , pamięć elastyczną (RAM) i pamięć stałą (HDD), może być wykonywany bezpośrednio – jeśli wyrażony jest w języku zrozumiałym dla danej maszyny lub pośrednio – gdy jest interpretowany przez inny program zwany interpretatorem. Programy komputerowe można zaklasyfikować według ich zastosowań. Wyróżnia się zatem:

- aplikacje użytkowe,

- systemy operacyjne,

- gry wideo,

- kompilatory i inne.

Programy wbudowane wewnątrz urządzeń określa się jako firmware.

4.3.Język programowania

Język programowania to zbiór zasad określających, kiedy ciąg symboli tworzy program (czyli ciąg symboli opisujący obliczenia) oraz jakie obliczenia opisuje. Podobnie jak języki naturalne, język programowania składa się ze zbiorów reguł syntaktycznych oraz semantycznych, które opisują, jak należy budować poprawne wyrażenia oraz jak komputer ma je rozumieć. Wiele języków programowania posiada pisemną specyfikację swojej składni oraz semantyki, lecz inne zdefiniowane są jedynie przez oficjalne implementacje.

Język programowania pozwala na precyzyjny zapis algorytmów oraz innych zadań, jakie komputer ma wykonać. W naszym przypadku użyjemy języka niskiego poziomu tzn. pracującego bezpośrednio na sprzęcie, będzie nim C, a dokładniej RobotC.

4.4.Serwomotor

Rys. 1. Serwomotor [1]

Serwomotor to silnik elektryczny wykorzystywany w urządzeniach w przypadku gdy siła ludzka nie jest wystarczalna bądź urządzenie działa automatyczne. W przypadku Lego Mindstorms NXT serwomotor jest silnikiem elektrycznym o mocy 9V, zawiera także zestaw metalowych przekładni, hamulec i czujnik obrotów. Montowany jest do kostki NXT do jednego z portów oznaczonym literami od A do C. Kabel zapewnia zasilanie i dwukierunkową komunikację. Pozwala to kostce NXT wydawać polecenia startu i zatrzymania swobodnego lub z hamowaniem, a także odczytywać aktualną prędkość i położenie oraz zliczać wykonane obroty.

4.5.Sensor

Sensor to urządzenie dostarczające informacji o pojawieniu się określonego bodźca, przekroczeniu pewnej wartości progowej lub o wartości rejestrowanej wielkości fizycznej. Przykładowe sensory to:

- sensor dotyku,

- sensor ultradźwiękowy,

- sensor koloru,

- sensor odległości,

- itp.

4.6.Sensor dotyku

Rys. 2. Sensor dotyku [1]

Sensor dotyku reaguje na dotyk poprzez naciśnięcie części ruchomej i tym samym zamknięciu obwodu, który wysyła sygnał do jednostki nadrzędnej. W przypadku robota Lego Mindstorms NXT wykrywa nacisk na umieszczony z przodu obudowy pomarańczowy przycisk i zwraca go kostce NXT poprzez połączenie analogowe. Przycisk posiada krzyżakowy otwór pozwalający na dołączenie osi i podobnych elementów lego w celu integracji z resztą konstrukcji.

4.7.Sensor koloru

Rys. 3. Sensor koloru [1]

Sensor koloru rozpoznaje kolory z zakresu RGB, potrafi także odtwarzać odpowiedni kolor z wyżej wymienionego przedziału. W celu rozpoznania koloru sensor wysyła pojedynczą wiązkę koloru na przedmiot i bada intensywność odbitej wiązki. Sensor koloru w Lego Mindstorms NXT wykorzystuje trzy różnokolorowe diody LED do oświetlenia powierzchni i mierzy intensywność odbitego światła dla każdego z nich. Dane te zwraca jako kolor w formacie RGB lub jako numer jednego z 18 kolorów podstawowych.

4.8.Dalmierz ultradźwiękowy

Rys. 4. Sensor Ultradźwiękowy [1]

Dalmierz ultradźwiękowy jest urządzeniem potrafiącym wydawać dźwięki i badać odległość między miejscem w którym się znajduje, a przeszkodą. W tym celu dalmierz wysyła sygnał ultradźwiękowy i bada czas z jakim sygnał wraca do urządzenia. W Lego Mindstorms NXT działa poprzez wysyłanie impulsów ultradźwiękowych i mierzenie czasu, po jakim wracają odbite od obiektu. Zintegrowana elektronika automatycznie dokonuje pomiaru i oblicza odległość w cm.

4.9.Kostka NXT

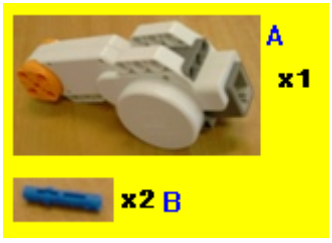
Rys. 5. Kostka NTX [1]

Kostka NXT to Inteligentna kostka sterująca pracą robota. Wyposażona jest w 32-bitowy procesor, 4 wejścia dla sensorów, 3 wyjścia dla serwomotorów, wyświetlacz LCD, głośnik i przyciski sterujące. Połączenie z komputerem zapewnia port USB lub bezprzewodowa technologia Bluetooth.

5. Budowa robota

Rys. 6. Potrzebne elementy

Krok 1.

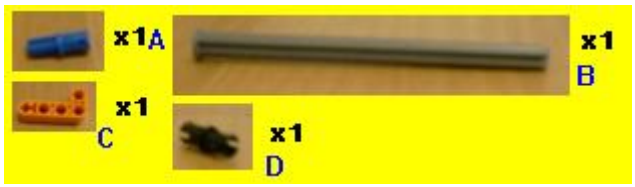


Rys. 7. Potrzebne elementy

Elementy B przełożyć przez element A tak jak na poniższym zdjęciu.

Rys. 8. Efekt pracy

Krok 2.

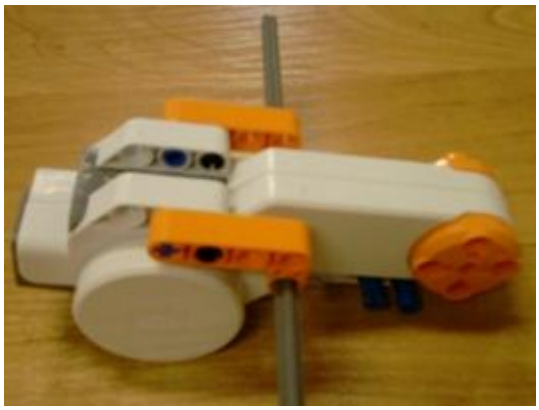


Rys. 9. Potrzebne elementy

Wszystkie elementy połączyć jak na zdjęciu poniżej.

Rys. 10. Instrukcja złożenia

Czynność powtórzyć, tworząc drugi taki sam element. Element z kroku 1 połączyć z powyższym elementem, tak jak na rysunku poniżej.



Rys. 11. Efekt pracy

Krok 3.

Rys. 12. Potrzebne elementy

Elementy połączyć z gotowym elementem z kroku 2, tak jak na poniższym zdjęciu.

Rys. 13. Instrukcja złożenia

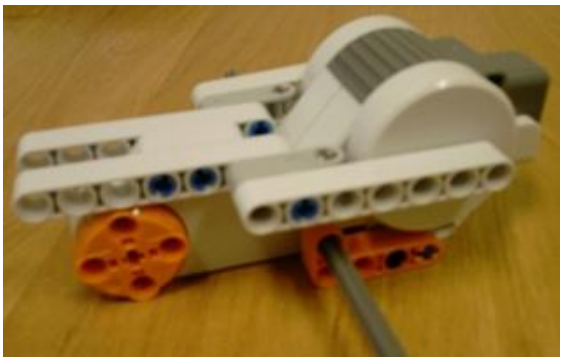
Krok 4.

Rys. 14. Potrzebne elementy

Wszystkie elementy połączyć jak na poniższym zdjęciu.

Rys. 15. Instrukcja złożenia

Powtarzamy czynność. Elementy należy połączyć jak na zdjęciu poniżej.

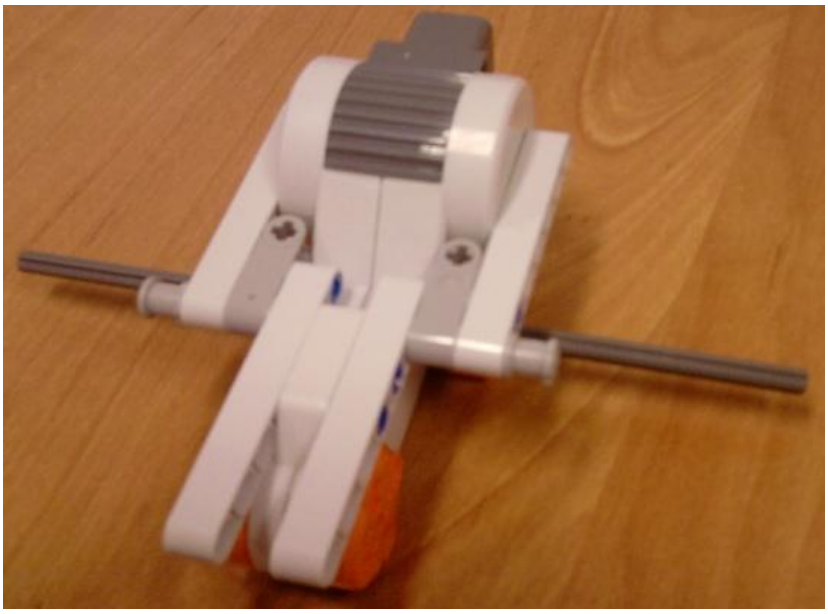


Rys. 16. Efekt pracy

Krok 5.

Rys. 17. Potrzebne elementy

Elementy należy połączyć z elementem z kroku 4, tak jak na zdjęciu poniżej.



Rys. 18. Instrukcja złożenia

Krok 6.

Rys. 19. Potrzebne elementy

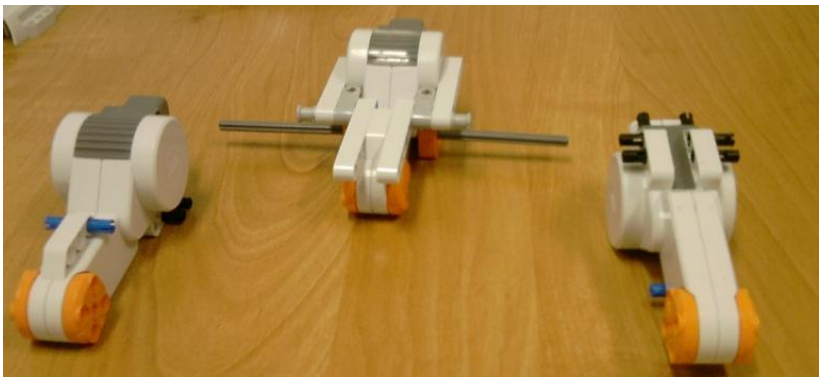
Wszystkie elementy należy złożyć jak na poniższym zdjęciu.

Rys. 20. Instrukcja złożenia

Krok 7.

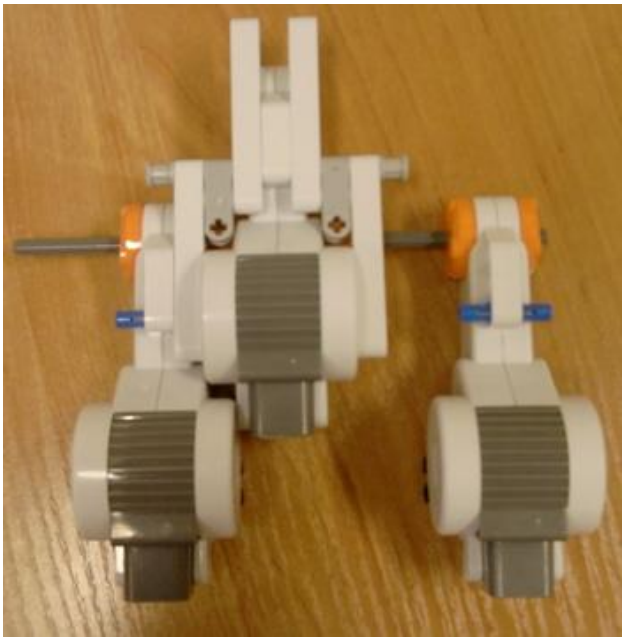
Należy powtórzyć krok 6.

Krok 8.



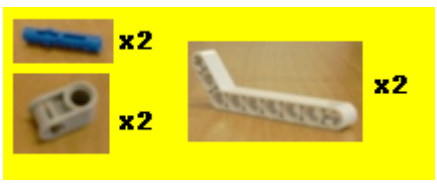
Rys. 21. Potrzebne elementy

Elementy przedstawione na zdjęciu powyżej, należy połączyć tak jak na zdjęciu poniżej:



Rys. 22. Instrukcja złożenia

Krok 9.

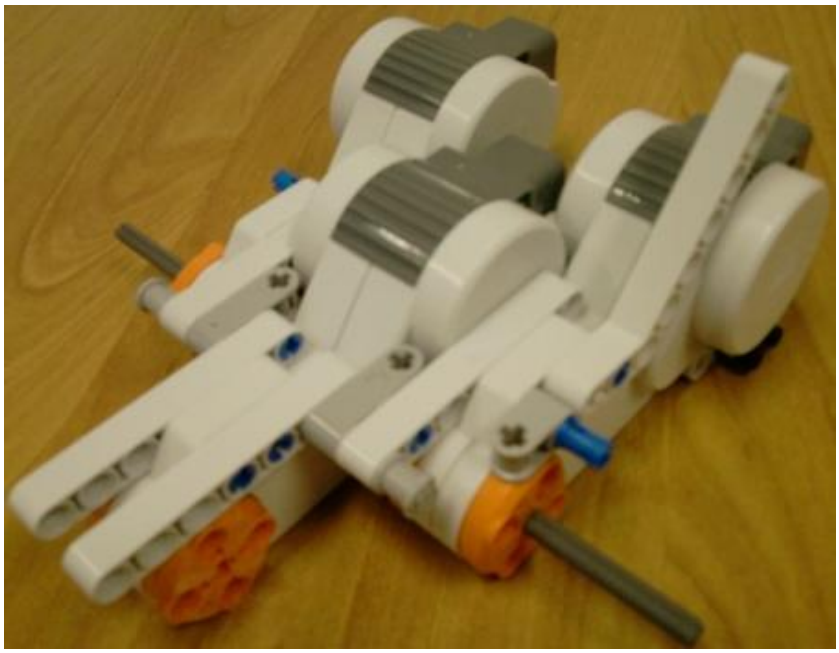


Rys. 23. Potrzebne elementy

Elementy należy połączyć tak jak na zdjęciu poniżej.

Rys. 24. Instrukcja złożenia

Całość połączyć jak na zdjęciu poniżej.

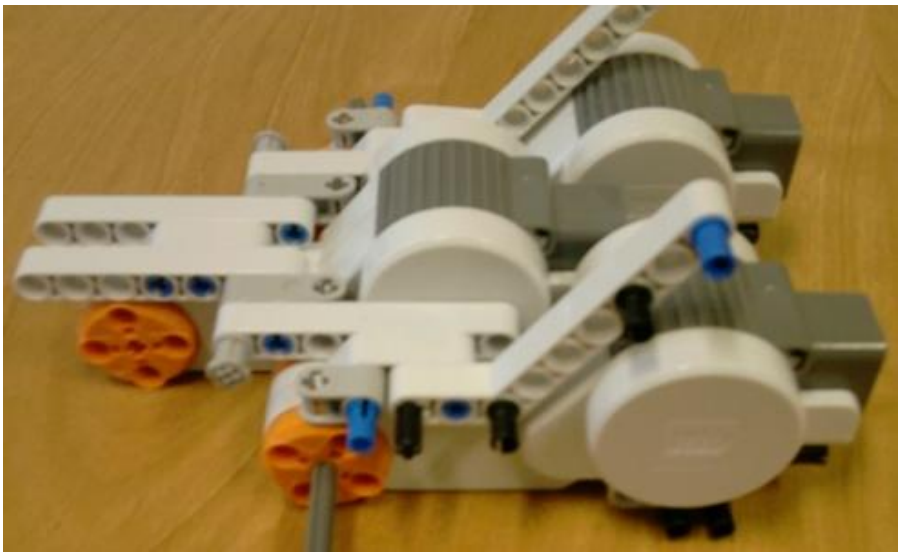


Rys. 25. Efekt pracy

Krok 10.

Rys. 26. Potrzebne elementy

Wszystkie elementy należy połączyć jak na zdjęciu poniżej, z obu stron identycznie.

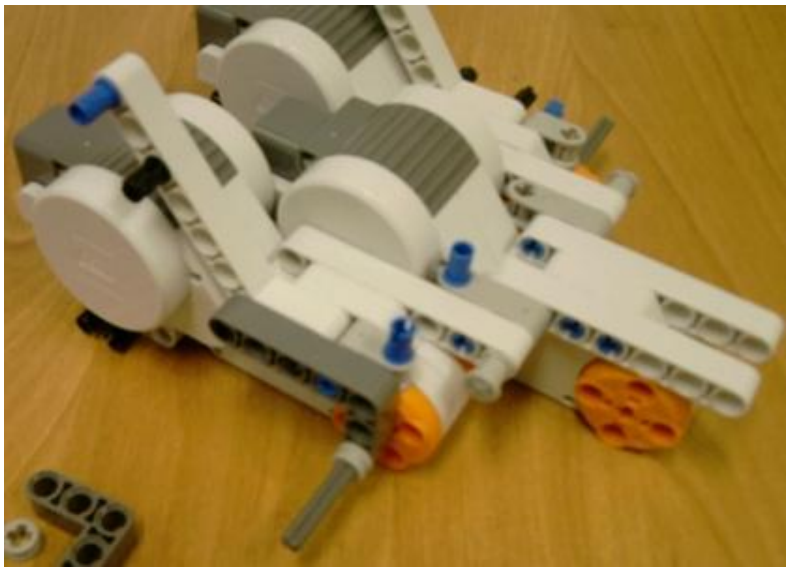


Rys. 27. Instrukcja złożenia

Krok 11.

Rys. 28. Potrzebne elementy

Wszystkie elementy należy połączyć jak na zdjęciu poniżej, z obu stron identycznie.

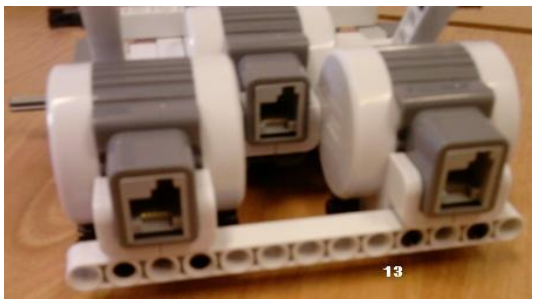


Rys. 29. Efekt pracy

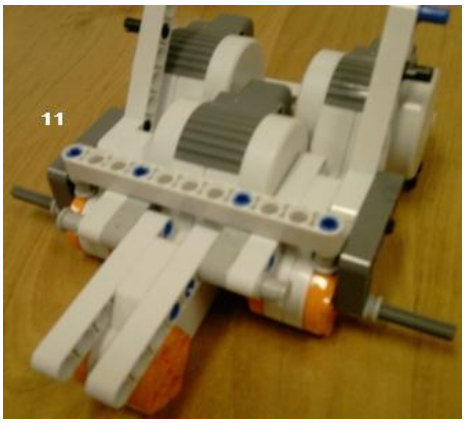
Krok 12.

Rys. 30. Potrzebne elementy

Elementy należy połączyć jak na zdjęciach poniżej.



Rys. 31. Instrukcja złożenia



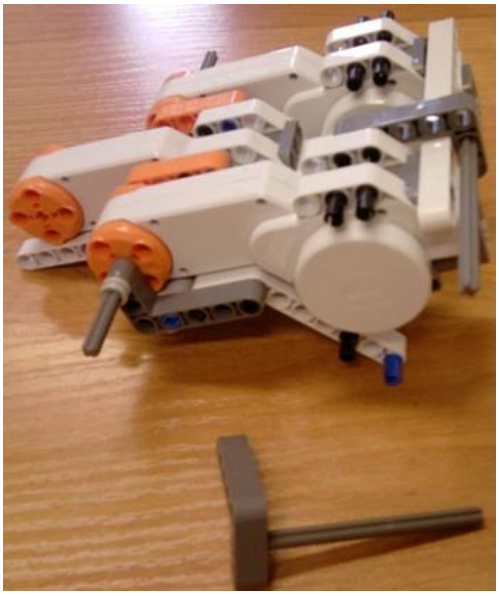
Rys. 32. Instrukcja złożenia

Krok 13.

Rys. 33. Potrzebne elementy

Wszystkie elementy połączyć jak na zdjęciach poniżej.

Rys. 34. Instrukcja złożenia



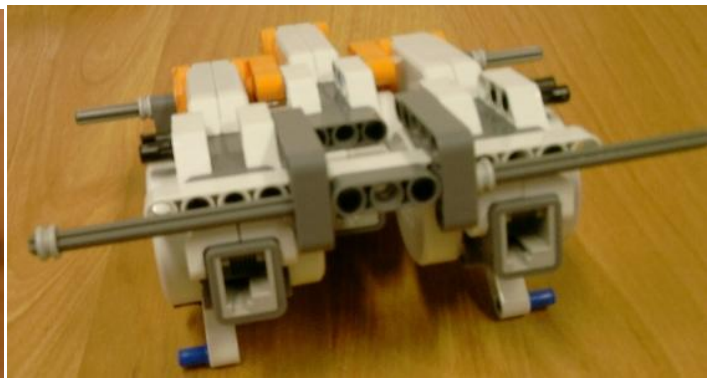
Rys. 35. Instrukcja złożenia

Krok 14.

Rys. 36. Potrzebne elementy

Wszystkie elementy połączyć jak na zdjęciach poniżej.

Rys. 37. Instrukcja złożenia

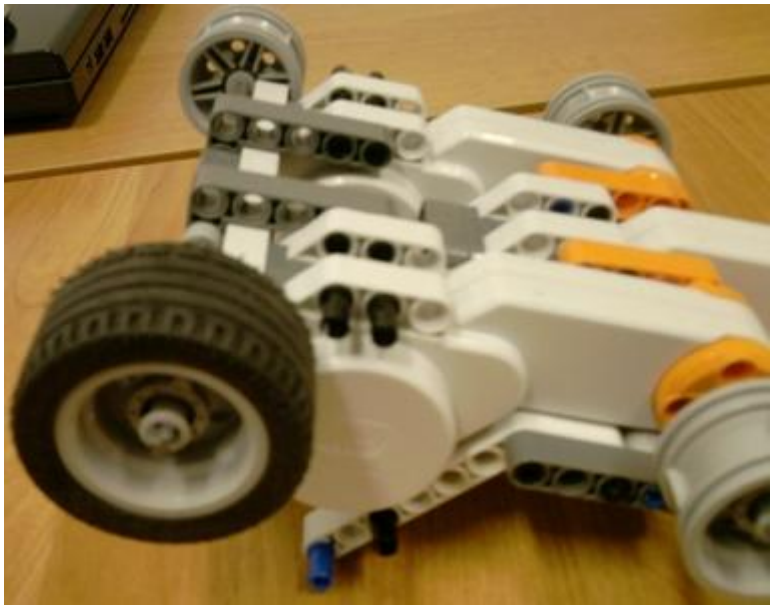


Rys. 38. Instrukcja złożenia

Krok 15.

Rys. 39. Potrzebne elementy

Wszystkie elementy połączyć jak na zdjęciu poniżej, z każdej strony identycznie.



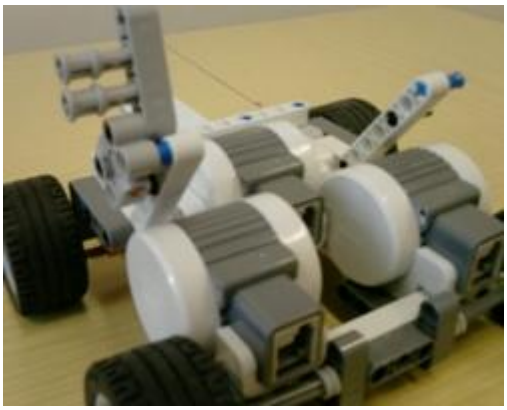
Rys. 40. Instrukcja złożenia

Krok 16.

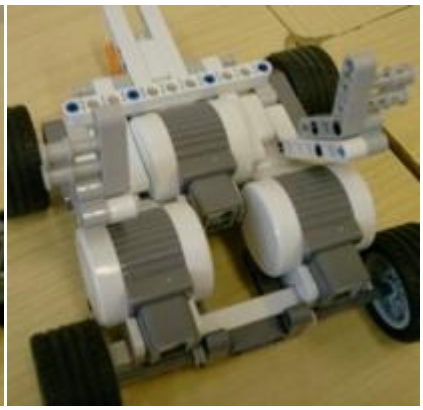
Rys. 41. Potrzebne elementy

Wszystkie elementy należy połączyć jak na zdjęciach poniżej, z obu stron identycznie.

Rys. 42. Instrukcja złożenia



Rys. 43. Instrukcja złożenia



Rys. 44. Instrukcja złożenia

Krok 17.

Rys. 45. Potrzebne elementy

Element dołączyć tak jak na poniższym zdjęciu, z obu stron identycznie.

Rys. 46. Instrukcja złożenia

Krok 18.

Rys. 47. Potrzebne elementy

Wszystkie elementy połączyć jak na zdjęciach poniżej, z obu stron identycznie.

Rys. 48. Instrukcja złożenia

Rys. 49. Instrukcja złożenia

Krok 19.

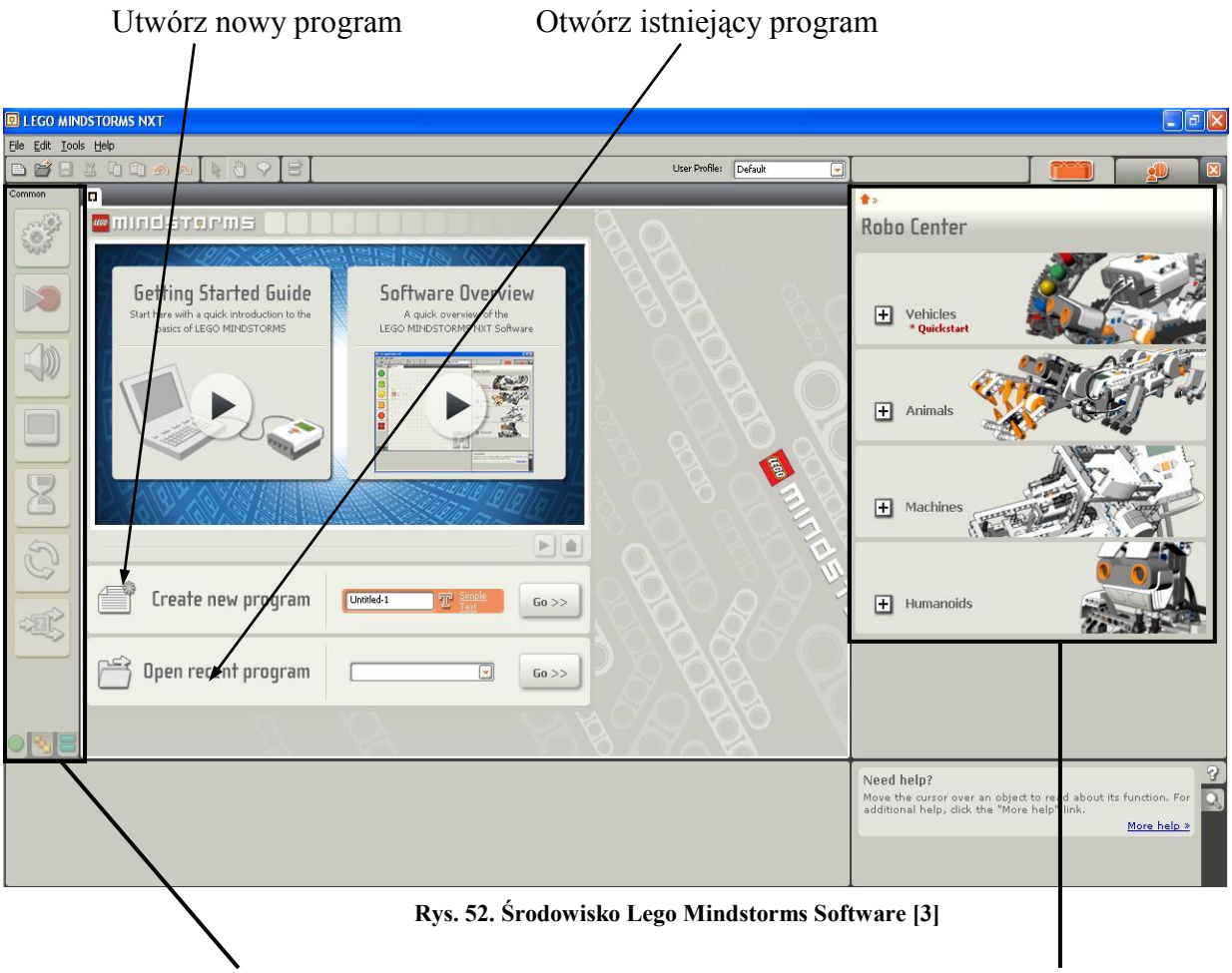
Rys. 50. Potrzebne elementy

Wszystkie kable połączyć jak na zdjęciu poniżej.

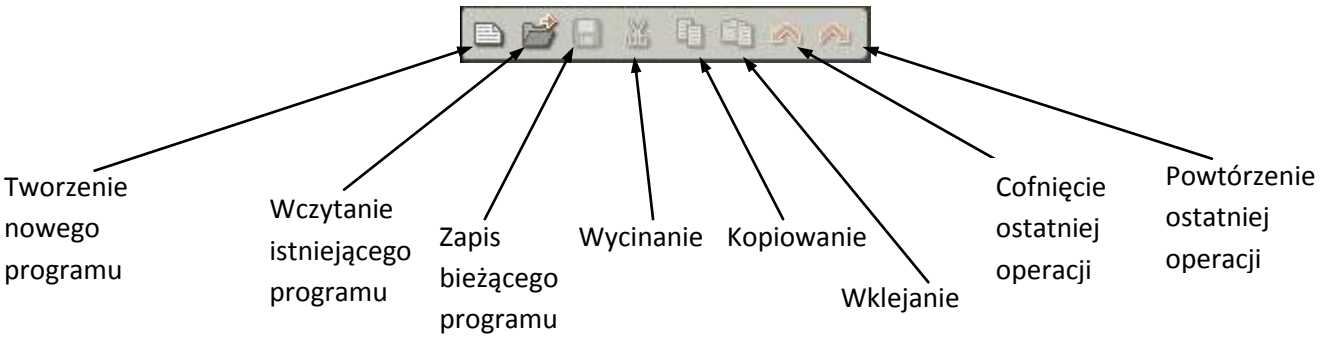
Rys. 51. Instrukcja złożenia

W razie jakich kol wiek problemów proszę zajrzeć do instrukcji „LEGO MINDSTORMS User Guide”.

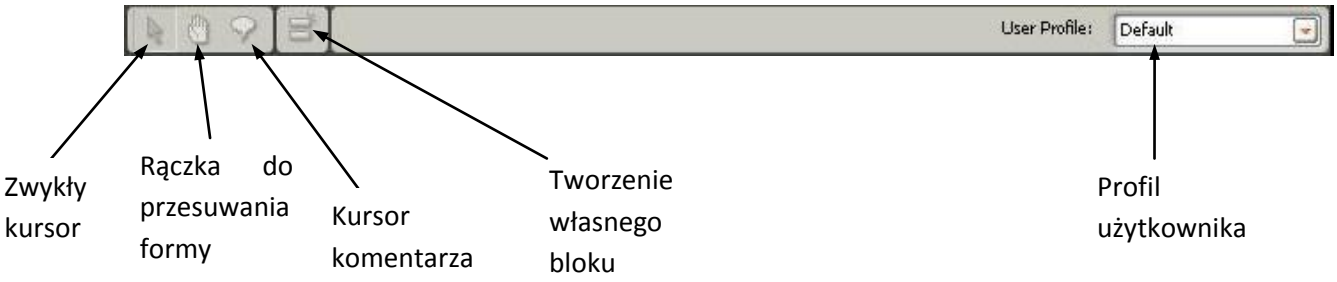
6. Środowisko Lego Mindstorms Software



Rys. 52. Środowisko Lego Mindstorms Software [3]



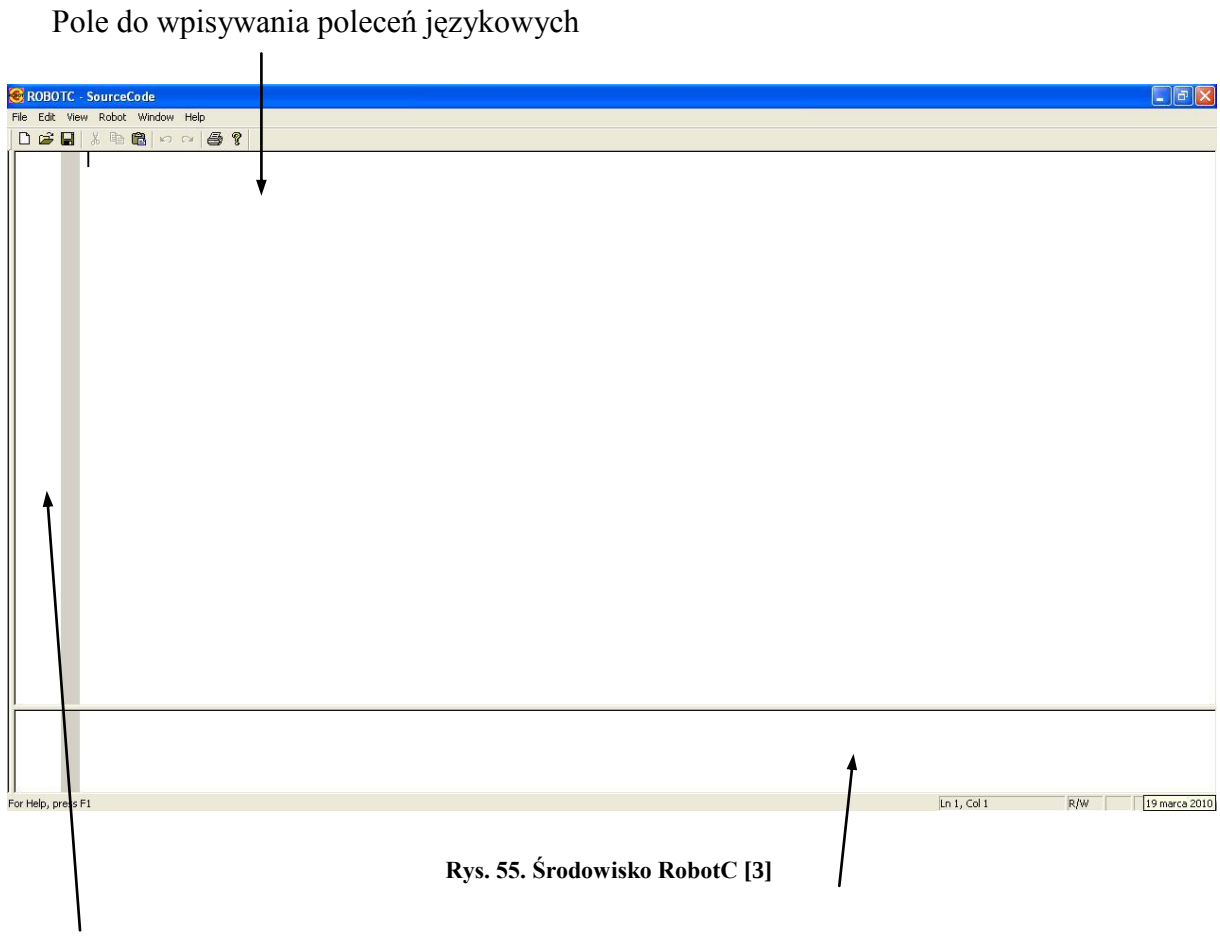
Rys. 53. Panel przycisków tworzenia i wczytywania programu [3]



Rys. 54. Panel przycisków myszki [3]

Więcej informacji znajdziesz w kursie dotyczącym programowania w środowisku Lego Mindstorms Software.

7. Środowisko RobotC



Rys. 55. Środowisko RobotC [3]

Reszta ikon jest standardowa, więc nie powinna nikomu sprawiać problemów.

8. Ćwiczenia praktyczne – algorytmika

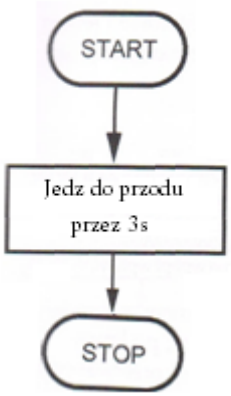
Aby móc poprawnie tworzyć programy, trzeba na początku wiedzieć, co i jak. Tym właśnie zajmuje się algorytmika – dzieleniem zadania na prostsze kroki i tworzenie ciągu ich wykonywania. Aby każdy czytelnik mógł bez problemu rozpocząć naukę pisania programów Lego Mindstorms Software bądź RobotC musi wiedzieć jak i co trzeba wykonać algorytmicznie aby rozwiązać przydzielone zadanie.

Oto kilka zadań pomocnych w zrozumieniu funkcjonowania robota.

8.1. Zadanie 1.

Zaprogramuj Robota aby jechał do przodu przez 3s.

Rozwiązanie:



Rys. 56. Algorytm

8.2. Zadanie 2.

Zaprogramuj Robota by jechał do przodu przez 3s, a następnie skręcił w lewo, operacja ta ma się powtórzyć 3 razy.

Rozwiązanie:

Rys. 57. Algorytm

8.3.Zadanie 3.

Zaprogramuj robota aby jechał do przodu aż nie napotka przeszkody. W przypadku jej napotkania ma pojechać do tyłu 1s.

Rozwiązanie:

Rys. 58. Algorytm

8.4. Zadanie 4.

Zaprogramuj robota aby jechał do przodu do napotkania obiektu na drodze w przypadku napotkania tego obiektu ma użyć serwomotora A i rozpocząć wszystko od początku. Ma w wykonać tą czynność 3 razy.

Rozwiązanie:

Rys. 59. Algorytm

8.5. Zadanie 5.

Zaprogramuj robota aby reagował na odpowiednie bodźce dotyk i ultradźwięk (reakcja od 20cm). W obu przypadkach ma pojechać do tyłu 2s.

Rozwiązanie:

Rys. 60. Algorytm

8.6.Pytania sprawdzające wiedzę - Algorytmika

1. Do rozpoczęcia algorytmu należy użyć:
    a. blok początkowy
    b. blok końcowy
    c. blok wykonawczy
2. Linia przepływu jest:
    a. skierowanym łącznikiem między dwoma blokami
    b. linią końca wiersza
    c. linią komentarza
3. Polecenia na danych wejściowych wykonuje się za pomocą:
    a. bloku komentarza
    b. bloku wykonawczego
    c. bloku wejścia \ wyjścia
4. Polecenie, które sprawdza dane wyrażenie i w zależności od jego wyniku zwraca prawda / fałsz to:
    a. pętla
    b. przełącznik
    c. polecenie warunkowe

5. Aby wykonać czynność określoną ilość razy należy użyć:
    a. przełącznika
    b. pętli
    c. instrukcji wejścia / wyjścia

8.7. Ćwiczenia do samodzielnego wykonania

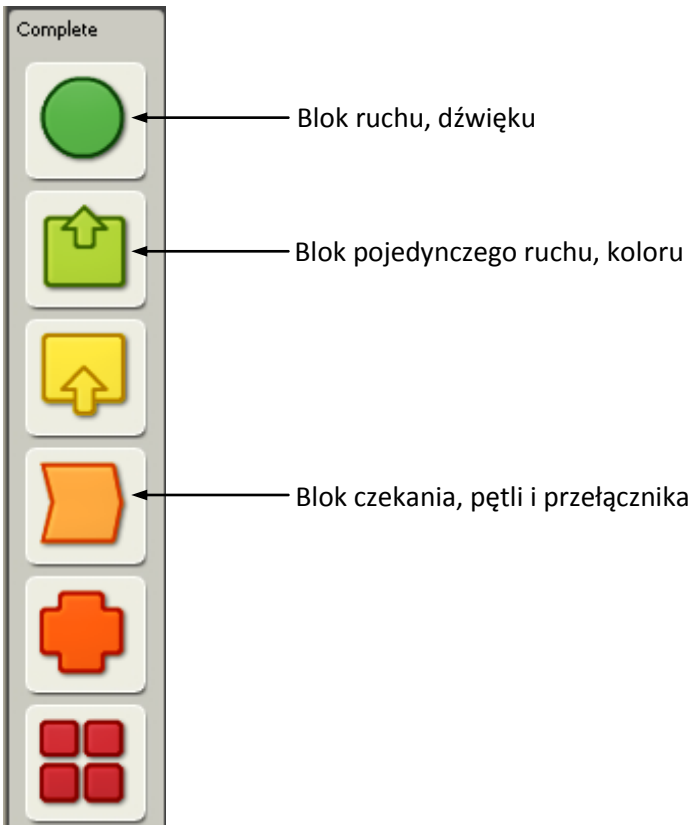
1. Zaprogramuj robota aby jechał do przodu, a po napotkaniu przeszkody w odległości 20cm ma obrócić się o 180^{o}.
2. Zaprogramuj robota aby jechał z taką prędkością, jaką odbiera poziom głos z czujnika dźwięku.
3. Zaprogramuj robota aby rozpoznawał w jakiej odległości znajdują się obiekty dookoła niego i podjechał do najbliższego z nich.
4. Zaprogramuj robota aby wykonał prosty układ taneczny np. obrót o 360^{o}, 1s w prawo, 1s w lewo i powtórzenie wszystkich czynności.
5. Zaprogramuj robota aby po dojechaniu do obiektu wydał dźwięk.

9. Programowanie w środowisku Lego Mindstorms Software

9.1.Podstawy

Aby rozpocząć programowanie w środowisku Lego Mindstorms Software należy poznać kilka podstawowych bloków.

Znajdują się one w buttonach zakładki Complete



Rys. 61. Panel przycisków akcji

Blok startowy

Od niego rozpoczyna się każdy program, ma 3 wyjścia, co oznacza, że robot może mieć do wykonywania 3 różne algorytmy.

Blok ruchu

Blok sterujący serwomotorami. W nim możemy ustalić, które serwomotory mają wykonać pracę, z jaką mocą, z jakim podziałem mocy między 2 serwomotory, kierunek obrotu i ilość powtórzeń.

Blok dźwięku

Blok odpowiadający za sterowanie dźwiękiem w robocie. W nim możemy ustawić rodzaj wydawanego dźwięku, ilość powtórzeń, głośność.

Blok pojedynczego ruchu

Blok sterujący 1 serwomotorem, Ma takie same własności, co blok ruchu.

Blok koloru

Blok sterujący sensorem koloru, może ustawić 1 z 3 kolorów (czerwony, niebieski, zielony). Należy w nim ustawić, do którego portu podłączyliśmy sensor. Możemy w nim ustalić czy sensor ma być włączony czy wyłączony.

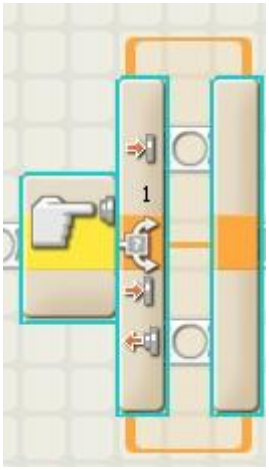
Blok czekania

W nim ustalamy, do jakiej akcji ma być wykonywany blok wcześniejszy. Mamy do wyboru opcje czasową bądź aktywację sensorem. Jeżeli wybierzemy sensor należy ustalić, na którym porcie ma być pobierana wartość sensora.

Pętla

Pętla służy do powtarzania niektórych czynności, określoną ilość razy. Można w niej ustawić ile razy ma być wykonywana lub jaka akcja ma ją kończyć.

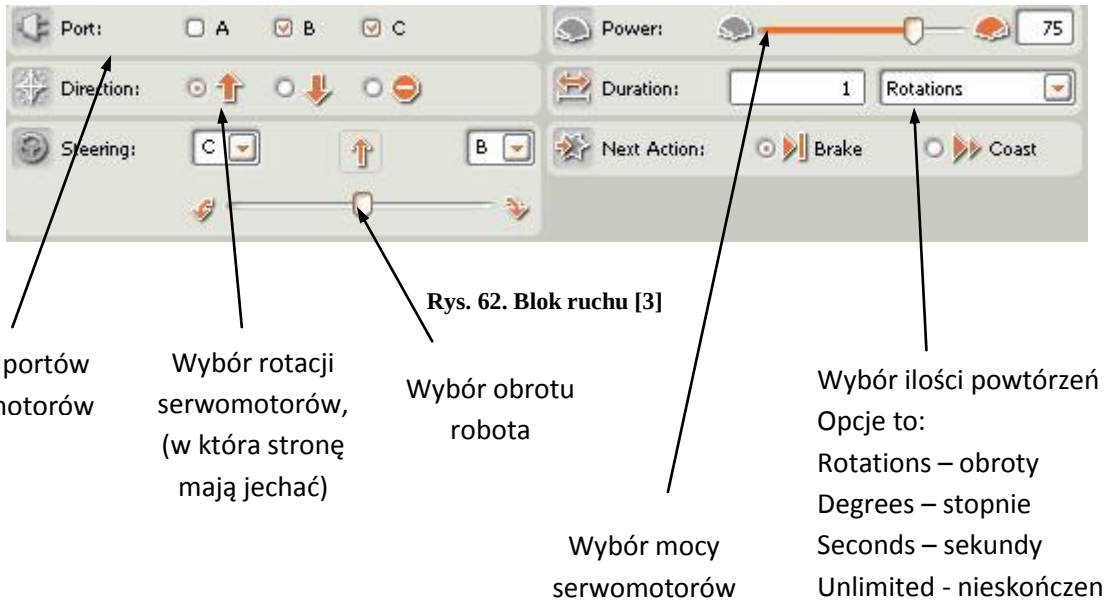
Przełącznik



Przełącznik służy do rozdzielania programu na 2 operacje ze względu na to, jaką wartość przyjmie parametr na początku przełącznika. Przy wyborze sensora należy pamiętać by ustawić odpowiedni port i czynności.

Znasz już podstawowe bloczki niezbędne do tworzenia programów. Teraz należy zapoznać się z ich własnościami.

Własności bloku ruchu



Rys. 62. Blok ruchu [3]

Wybór portów serwomotorów

Wybór rotacji serwomotorów, (w która stronę mają jechać)

Wybór obrotu robota

Wybór mocy serwomotorów

Wybór ilości powtórzeń
Opcje to:
Rotations – obroty
Degrees – stopnie
Seconds – sekundy
Unlimited - nieskończenie

Własności bloku dźwięku

Rys. 63. Blok dźwięku [3]

Rodzaj odtwarzanego dźwięku

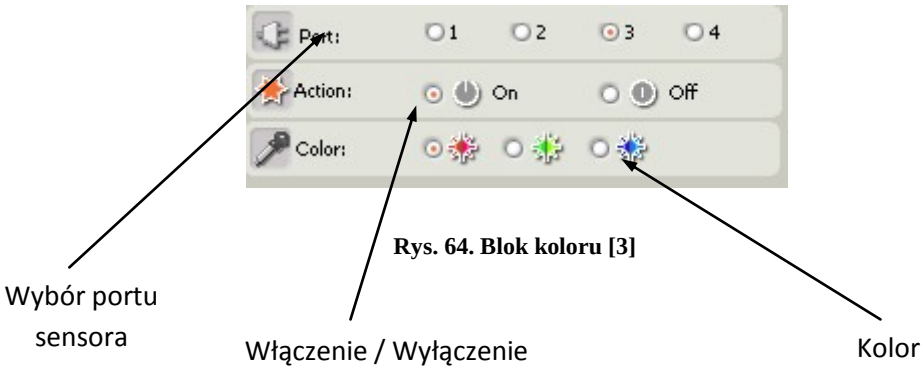
Włączenie / Wyłączenie

Dostępne dźwięki

Głośność

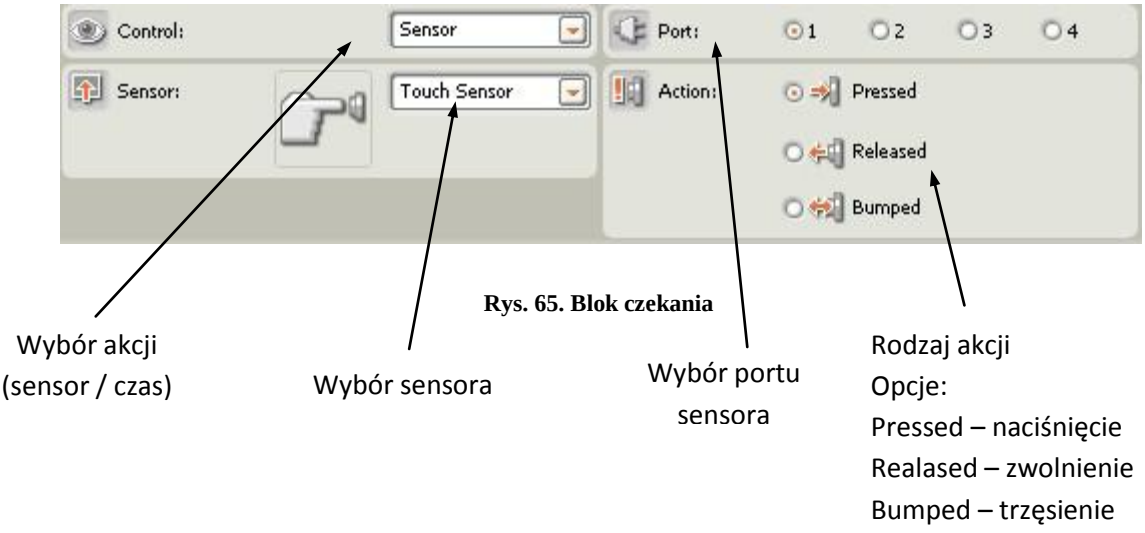
Powtarzalność

Własności bloku koloru



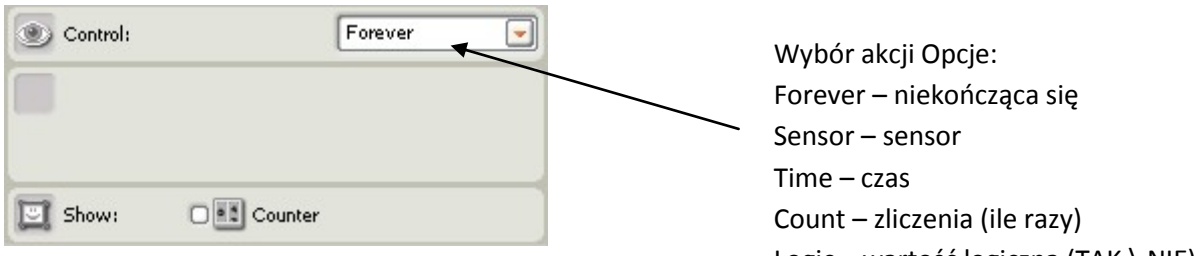
Rys. 64. Blok koloru [3]

Własności bloku czekania



Rys. 65. Blok czekania

Własności pętli



Rys. 66. Pętla

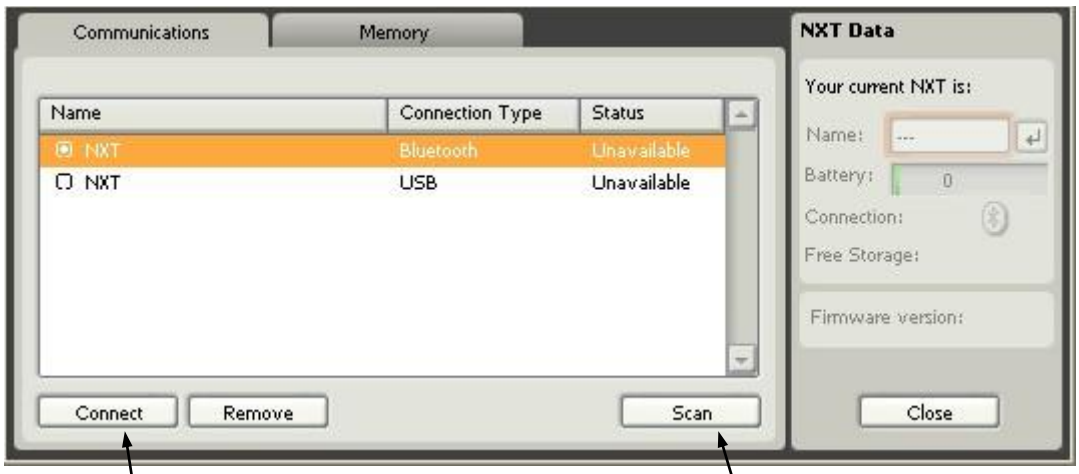
9.1.1. Kompilacja

Aby skompilować program należy:

1. Podłączyć kabel USB / skonfigurować połączenie bluetooth komputera z robotem.
2. Teraz należy połączyć się kompilatorem z robotem:

Rys. 67. Łączenie z kostką [3]

Wybieramy zaznaczoną opcję.



Rys. 68. Komunikacja z kostką [3]

Połączenie

Skanowanie w poszukiwaniu urządzenia

Po udanym połączeniu w kolumnie Status powinien wyświetlić się nam napis, Connected.

3. Naciśnij przycisk kompiluj w kompilatorze.

Kompilacja wybranego kawałka kodu

Kompilacja i uruchomienie programu

Sama kompilacja

Zatrzymanie pracy robota

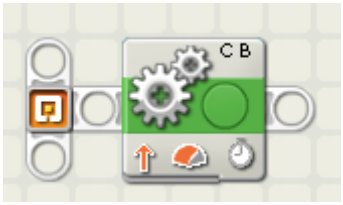
Rys. 69. Kompilacja [3]

9.2.Programy sterujące ruchem robota

9.2.1. Zadanie 1.

Zaprogramuj robota tak, aby jechał do przodu 3s.

Rozwiązanie:



Rys. 70. Wykonanie [3]

We właściwościach bloku ruchu ustawiamy Duration na Seconds i wpisujemy wartość 3.

9.2.2. Zadanie 2.

Zaprogramuj robota, aby jechał do tyłu 3s.

Rozwiązanie:

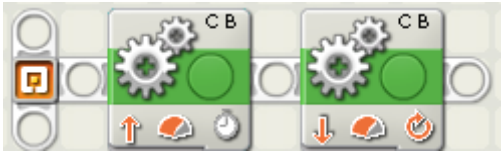
Rys. 71. Wykonanie [3]

We właściwościach bloku ruchu ustawiamy Duration na Seconds i wpisujemy wartość 3.

9.2.3. Zadanie 3.

Zaprogramuj robota, aby jechał do przodu 5s, po czym cofną się do tyłu 3 obroty.

Rozwiązanie:



Rys. 72. Wykonanie [3]

Opis ustawienia pierwszego nie powinien sprawiać problemów, natomiast w drugim w Duration wybieramy Rotations i ustawiamy na 3.

9.2.4. Zadanie 4.

Zaprogramuj robota, aby skręcił w lewo o 45^{o}.

Rozwiązanie:

Rys. 73. Wykonanie [3]

We właściwościach ruchu należy ustawiamy Steering maksymalnie w lewo i ustawiamy Rotations na 2 obroty. Oczywiście robot, może nie wykonać pełnego ruchu bądź zrobić za duży obrót spowodowane jest różnością nawierzchni, na której robot wykonuje ruch.

9.2.5. Zadanie 5.

Zaprogramuj robota, aby skręcił w prawo o 90^{o}.

Rozwiązanie:

Rys. 74. Wykonanie [3]

Ustawiamy Steering maksymalnie w prawo i ustawiamy Rotations na 4 obroty.

9.2.6. Ćwiczenia do samodzielnego wykonania

1. Zaprogramuj robota tak, aby jechał do przodu 10 obrotów.
2. Zaprogramuj robota tak, aby jechał do tyłu 10 obrotów.
3. Zaprogramuj robota tak, aby skręcił o 135^{o}.
4. Zaprogramuj robota tak, aby jechał do przodu 5s, skręcił o 180 ^{o} i wrócił do miejsca startu.
5. Zaprogramuj robota tak, aby zrobił 1 okrążenie po obwodzie kwadratu.

9.3.Sensory

Do poniższych ćwiczeń trzeba podłączyć odpowiednie sensory do odpowiednich portów!

9.3.1. Zadanie 1.

Zaprogramuj robota, aby jechał do przodu aż napotka coś na swojej drodze, wykorzystaj do tego sensor dotyku.

Rozwiązanie:

Rys. 75. Wykonanie [3]

We własnościach bloku ruchu ustawiamy Duration na Umlimited, po czym wybieramy blok czekania, opcja touch sensor (sensor dotyku) jest ustawiona automatycznie, wybieramy tylko odpowiedni port, do którego jest podłączony sensor i program po dostaniu wartości z sensora przerwie pracę operacji wcześniejszej i przejdzie dalej, czyli zakończy swoją pracę.

9.3.2. Zadanie 2

Zaprogramuj robota tak, aby jechał do przodu, po napotkaniu przeszkody (10cm od robota) cofną się o 2 obroty i zakończył swoją pracę. Do wykrywania przeszkód użyj sensora ultradźwiękowego.

Rozwiązanie:

Rys. 76. Wykonanie [3]

Własności bloku ruchu są takie same jak w zadaniu wcześniejszym, w bloku czekania wybieramy Sensor na Ultrasonic Sensor i ustawiamy Distance na 10cm, musimy także ustawić odpowiedni znak w tym przypadku mniejszości. Ustawienie opcji ostatniego sensora ruchu także nie powinno stanowić problemu.

9.3.3. Zadanie 3.

Zaprogramuj robota tak, aby po zlokalizowaniu w odległości 10cm obiektu zaświecił sensorem koloru na czerwono i wydał dźwięk.

Rozwiązanie:

Rys. 77. Wykonanie [3]

Ustawiamy własności ruchu takie same jak we ćwiczeniu wcześniejszym, we właściwościach bloku koloru ustawiamy na czerwony, a we właściwościach bloku dźwięku wybieramy jeden z przykładowych dźwięków. Należy pamiętać, aby porty bloków zgadzały się z fizycznymi podłączeniami sensorów do robota.

9.3.4. Zadanie 4.

Zaprogramuj robota, aby jechał przez 5s (przy użyciu bloku czekania), po czym obrócił się i wrócił na swoje miejsce.

Rozwiązanie:

Rys. 78. Wykonanie [3]

Właściwości bloków ruchu nie powinny stanowić problemu, we właściwościach bloku czekania wybieramy Control i wstawiamy wartość 5 w Until.

9.3.5. Zadanie 5.

Zaprogramuj robota, aby jechał do przodu dopóki jedzie po czarnym polu.

Rozwiązanie:

Rys. 79. Wykonanie [3]

We właściwościach bloku czekania ustawiamy Sensor na Color Sensor, wybieramy Range na kolo czarny, upewniamy się, że jest zaznaczona opcja Inside Range, czyli w zakresie tego koloru i ustawiamy odpowiedni dla nas port.

9.3.6. Ćwiczenia do samodzielnego wykonania

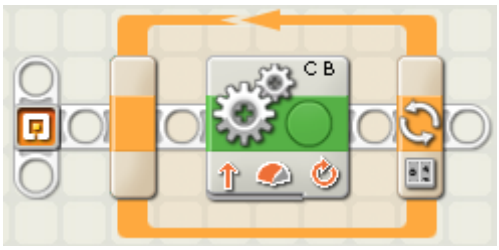
1. Zaprogramuj robota tak, aby jechał do dotknięcia obiektu, po czym ma cofnąć się, obrócił o 45 ^{o}, po czym jechał do napotkania kolejnego obiektu.
2. Zaprogramuj robota tak, aby za pomocą sensora koloru świecił przez 1s po kolei każdym kolorem
3. Zaprogramuj robota tak, aby wykrywał, w jakiej odległości jest obiekt naprzeciwko niego i jeżeli jest to odległość mniejsza niż 20cm ma wydać dźwięk.
4. Zaprogramuj robota tak, aby jechał do przodu 2s, po czym wydał dźwięk i zaświecił czerwoną diodą na sensorze koloru.
5. Zaprogramuj robota tak, aby jechał do tyłu 2s, po czym wydał dźwięk i zaświecił niebieską diodą na sensorze koloru.

9.4.Pętle i przełączniki

9.4.1. Zadanie 1.

Zaprogramuj robota tak, aby przy użyciu pętli pojechał do przodu o 5 obrotów.

Rozwiązanie:

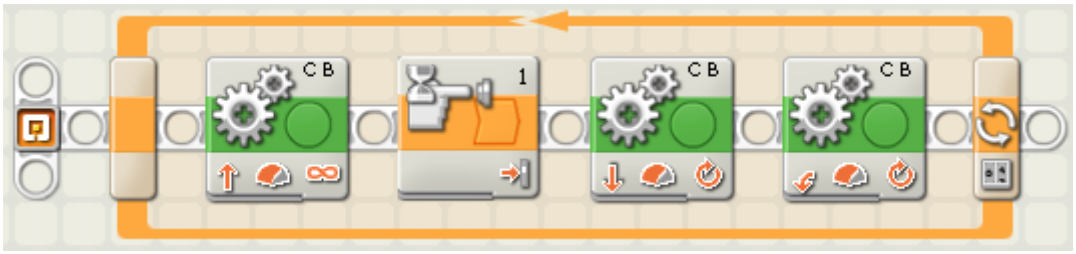


Rys. 80. Wykonanie [3]

Ustawiamy we właściwościach ruchu Rotations i ilość obrotów na 1. We właściwościach pętli wybieramy Control na Count (zliczenia), po czym ustawiamy ilość powótrzen (Count) na 5.

9.4.2. Zadanie 2.

Zaprogramuj robota tak, aby jechał do napotkania przeszkody (przy użyciu sensora dotyka), po czym cofną się i obrócił o 45^{o}, a całość operacji powtórzył 3 razy. Rozwiązanie:



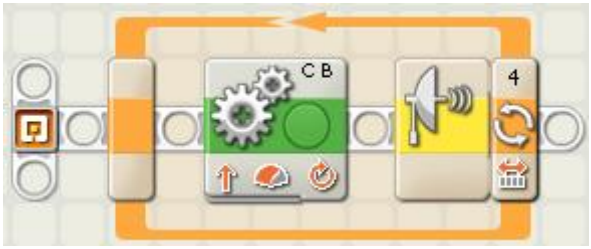
Rys. 81. Wykonanie

Wnętrze pętli nie powinno już stanowić problemu, całość ustawiamy na wykonywanie 3 razy tak jak było to w zadaniu 1.

9.4.3. Zadanie 3.

Zaprogramuj robota tak, aby jechał dopóki nie zostanie napotkany obiekt w odległości 10cm, przy użyciu pętli.

Rozwiązanie:



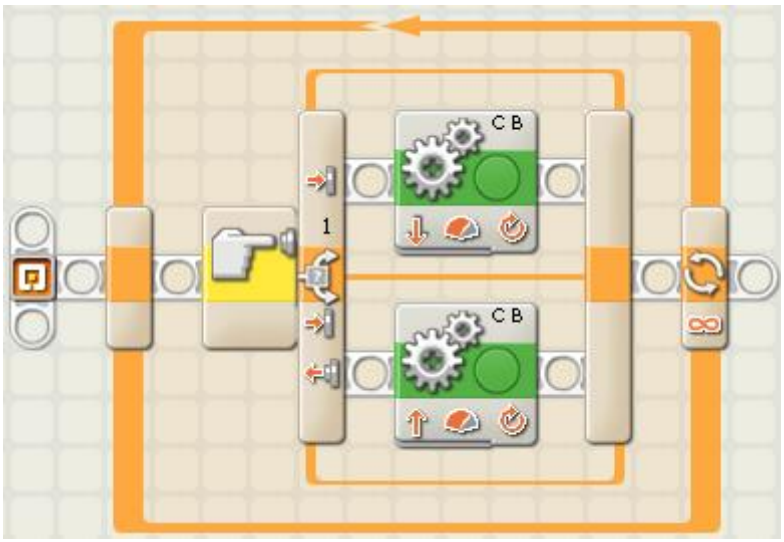
Rys. 82. Wykonanie

Należy ustawił właściwości bloku ruchu na 1 obrót, a właściwości pętli ustawić następująco: Opcje Control ustawić na wartość Sensor, Opcje Sensor ustawić na Ultrasonic Sensor i Distance ustawić na 10cm.

9.4.4. Zadanie 4.

Zaprogramuj robota tak, aby w zależności od tego czy dotyka jakiegoś obiektu czy nie jechał do przodu lub do tyłu.

Rozwiązanie:



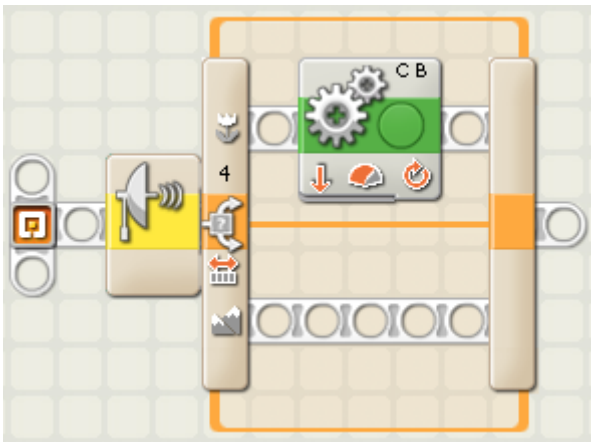
Rys. 83. Wykonanie [3]

Wstawiamy w pętle przełącznik, w którym automatycznie jest ustawiony sensor dotyku, w opcji Pressed(pierwsza z góry) blok ruchu ustawiony na jazdę do tyłu. W opcji Released(druga z góry) wstawiamy blok ruchu ustawiony na jazdę do przodu.

9.4.5. Zadanie 5.

Zaprogramuj robota tak, aby jechał do tyłu, jeżeli jest w odległości mniejszej niż 20cm, w przeciwnym wypadku ma nie robić nic.

Rozwiązanie:



Rys. 84. Wykonanie [3]

Przełącznik ustawiamy na tak aby pobierał wartości z sensora ultradźwiękowego w Distance ustawiamy na 20 cm.

9.4.6. Ćwiczenia do samodzielnego wykonania

1. Zaprogramuj robota tak, aby jechał do przodu 6 obrotów przy pomocy pętli
2. Zaprogramuj robota tak, aby jechał do przodu 1s, po czym skręcał o 45^{o}, wszystko ma być powtórzone 6 razy.
3. Zaprogramuj robota tak, aby jechał do tyłu 1s, po czym zaświecił diodą na zielono, wszystko ma być powtórzone 3 razy.
4. Zaprogramuj robota tak, aby jechał do przodu 1s, później przy pomocy przełącznika sprawdzał czy nie stoi w odległości do 20cm od niego jakiś obiekt. Jeżeli stoi, to robot ma się cofnąć o 1s w innym przypadku ma nie robić nic.
5. Zaprogramuj robota tak, aby sprawdził, w jakiej odległości od niego jest obiekt, jeżeli w odległości do 20cm ma zaświecić czerwoną diodą, w przeciwnym wypadku ma zaświecić zieloną diodą.

9.5.Pytania sprawdzające wiedzę – Lego Mindstorms Software

1) Aby robot użył tylko jednego serwomotoru należy użyć bloku:
   a) ruchu
   b) dźwięku
   c) pojedynczego ruchu
2) Aby robot skręcił należy ustawić wartość:
   a) duration
   b) distance
   c) steering
3) Aby ustawić świecenie się czerwonej diody w bloku koloru należy ustawić:
   a) duration
   b) color
   c) control
4) Jeżeli chcemy by robot wykonywał pewną czynność przez określony czas po tej czynności musimy ustawić blok:
   a) czekania
   b) dźwięku
   c) pętli
5) Przełącznik służy do:
   a) podejmowania pewnych decyzji w zależności od pewnego warunku
   b) powtarzania pewnych bloków
   c) wykonywania ruchu robotem

10.Programowanie w środowisku RobotC

10.1. Podstawy

Aby zacząć coś programować trzeba poznać składnie języka, oto kilka podstawowych funkcji, które czytelnik musi znać.

```
main();
```

Odpowiednikiem funkcji main() w RobotC jest task main(), więc szkielet programu jest następujący:

```
task main(){

}

motor[];
```

Jednym z podstawowych funkcji robota jest poruszanie swoimi serwomotorami, służy do tego funkcja motor[] o następującej składni:

```
motor[motorA] = 100;
```

Nazwa używanego serwomotoru, składa się z motor + nazwa portu (A-C)

Siła używana do wprawienia w ruch serwomotor

!Gdy ustawiamy siłę serwomotoru, nie ustawiamy ile obrotów ma on wykonać czy ile sekund jechać tylko, jaką ma pobierać moc z baterii do poruszania się, pamiętaj o tym.

```
wait1Msec();
```

Wiemy już jak poruszać poszczególnymi serwomotorami robota, lecz gdybyśmy skompilowali i uruchomili program ku naszemu zaskoczeniu nie stałoby się nic, jest to logiczne, ponieważ stworzyliśmy program, w którym serwomotor wprawiany jest w ruch, lecz program po naszym wywołaniu funkcji motor[] Idze dalej i widząc koniec funkcji task main() sam automatycznie kończy się, co jest skutkiem tego, że robot nie wykonuje akcji. Aby wstrzymać pracę programu i doprowadzić do uruchomienia serwomotoru twórcy RobotC stworzyli funkcje wait1Msec() o następującej składni:

```
wait1Msec(1000);
```

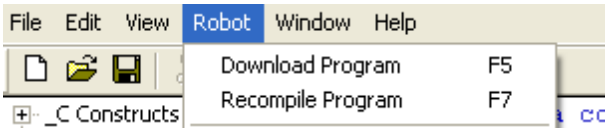
Liczba mikrosekund, jaką ma program czekać z przejściem do następnej linii kodu

1sec = 1000Msec, więc jeżeli chcemy, aby nasz robot jechał do przodu przez 1s musimy wpisać, 1000 jako argument funkcji.

Tworzenie funkcji jest identyczne jak w zwykłym C, zakładając, że czytelnik zna C nie opisuje tego w tym kursie.

Kompilacja

Aby skompilować jakikolwiek program należy wejść w zakładkę Robot, z menu programu



Rys. 85. Kompilacja [3]

Twórcy dali nam do wyboru 2 opcje:

- Download Program (jeżeli jeszcze nie skompilowaliśmy programu będzie widniało, jako Compile and Download Program)

- Compile Program (jeżeli skompilowaliśmy już program będzie widniało jako Recompile Program)

Download Program służy do skompilowania naszego programu i wysłania go do kostki NXT, jeżeli chcemy przesłać program do kostki NXT przez Bluetooth połączenie kostki z komputerem musi być już skonfigurowane i aktywne, podczas wysyłania program sam sprawdzi czy nie ma aktywnego połączenia. W przeciwnym wypadku wybierze połączenie przez USB, do wywołania ten akcji można użyć także skrótu F5.

Compile Program – kompiluje program bez wysyłania go do kostki NXT. Dobry sposób na sprawdzenie poprawności programu i programowanie bez posiadania kostki NXT, do wywołania ten akcji można użyć także skrótu F7.

!Podczas pierwszej kompilacji program poprosi nas o zapis pliku na dysku.

10.1.1. Ćwiczenia praktyczne – obsługa serwomotorów

Znasz już podstawowe funkcje do obsługi serwomotorów, więc czas teraz napisać jakiś program z wykorzystaniem tych funkcji.

10.1.1.1. Zadanie 1.

Zaprogramuj robota, aby pojechał do przodu przez 1s.

Rozwiązanie:

```
task main()

{

motor[motorC] = 100;

motor[motorB] = 100;

wait1Msec(1000);

}
```

Program jest banalnie prosty, wprawiamy w ruch na serwomotory (B i C, do których powinny być podłączone koła) w ten sposób robot ruszy do przodu i będzie jechał przez 1s.

10.1.1.2. Zadanie 2.

Zaprogramuj robota, aby pojechał do tyłu przez 1s.

Rozwiązanie:

```
task main()

{

motor[motorC] = -100;

motor[motorB] = -100;

wait1Msec(1000);

}
```

Program podobny do wcześniejszego, jedyną różnią jest, że wartości siły jakie przyjmują serwomotory są ujemne co oznacza, że będą kręcić się w przeciwnym kierunku do ruchu wskazówek zegara (do tyłu).

10.1.1.3. Zadanie 3.

Zaprogramuj robota, aby obrócił się o 45^{o}.

Rozwiązanie:

```
task main()

{

motor[motorC] = -50;

motor[motorB] = 50;

wait1Msec(800);

}
```

Program spowoduje, że robot obróci się w lewo (oczywiście jeżeli serwomotory są podłączone we właściwe porty). Oczywiście ustawienie innych sił mocy serwomotorów i czasu ich działania zmieni kąt obrotu, lecz ten aspekt zostawiam już ciekawości czytelnika.

10.1.1.4. Zadanie 4.

Zaprogramuj robota tak, aby pojechał do przodu 2s, po czym skręcił w lewo i przejechał do przodu kolejne 2s.

Rozwiązanie:

```
task main()

{

// jazda prosto

motor[motorC] = 100;

motor[motorB] = -100;

wait1Msec(2000);

// skręcenie w lewo

motor[motorC] = -50;

motor[motorB] = 50;

wait1Msec(800);
```

```
// jazda prosto

motor[motorC] = 100;

motor[motorB] = 100;

wait1Msec(2000);

// zatrzymanie się

motor[motorC] = 0;

motor[motorB] = 0;

}
```

Jak widać ten program jest już bardziej rozbudowany niż poprzednie, dla zwiększenia czytelności wstawiłem komentarze pomiędzy poszczególnymi działaniami robota. W ostatnich wierszach ustawiłem wartości mocy serwomotorów na 0 aby zaprezentować jak w czasie programu można zatrzymać robota, nie jest to wymagane w tym przykładzie ale może przydać się w innych bardziej rozbudowanych.

10.1.1.5. Zadanie 5.

Zaprogramuj robota tak, aby pojechał do przodu 1s po czym skręcił w prawo. Ma wykonać te ruchy 4 razy (pętla).

Rozwiązanie:

```
task main()

{

for(int i=0;i<4;++i)

{

motor[motorC] = 100;

motor[motorB] = 100;

wait1Msec(1000);

motor[motorC] = 50;

motor[motorB] = -50;

wait1Msec(800);
```

```
}

}
```

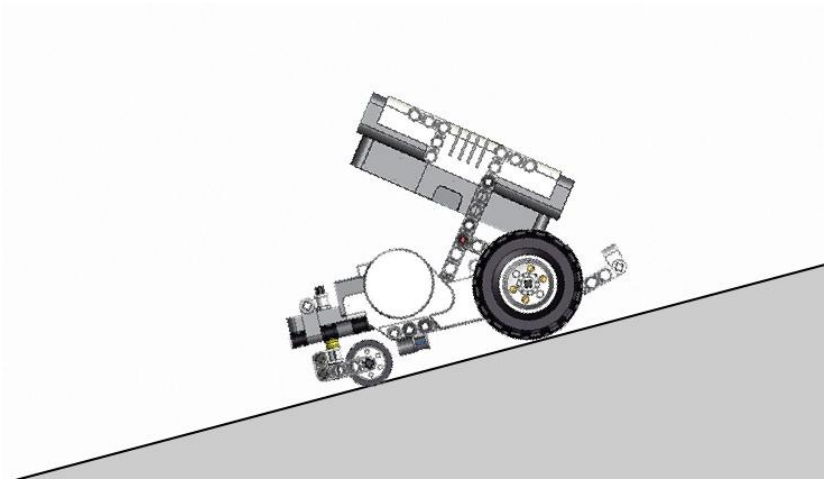
Program przedstawia użycie prostej pętli for. W konsekwencji robot przejedzie po torze kwadratu.

10.1.2. Ćwiczenia do samodzielnego wykonania

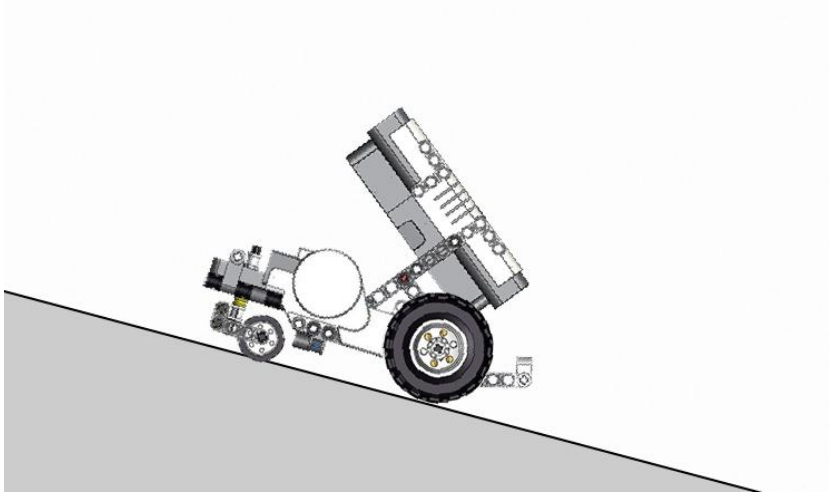
1) Zaprogramuj robota, aby pojechał do przodu 2s po czym ruszył do tyłu 1,5s.
2) Zaprogramuj robota tak, aby poruszał się po torze prostokąta.
3) Zaprogramuj robota, aby pojechał do przodu 3s po czym obrócił się i wrócił na swoje miejsce.
4) Zaprogramuj robota, aby wykonał prostą sekwencje taneczna:
   a) skręt w lewo, jazda prosto 2s,
   b) obrót o 405^{o} w lewo, jazda prosto 4s,
   c) obrót o 460^{o} w prawo, jazda prosto 2s.
   d) obrót o 45^{o} w prawo.
5) Zaprogramuj własną choreografie taneczną dla robota, która musi składać się przynajmniej z 6 ruchów.

10.1.3. *Zaawansowane sterowanie serwomotorami

Aby przejść do ten części kursu musisz zapoznać się z pojęciem regulator PID. PID czyli (ang. proportional-integral-derivative controller – regulator proporcjonalno-całkująco-różniczkujący) jest to narzędzie, które reguluje moc używanych serwomotorów, aby robot mógł się poruszać o taką samą odległość nie zależnie od terenu i kąta po którym jeździ. Ilustracje powinny pomóc każdemu w zrozumieniu tego mechanizmu.



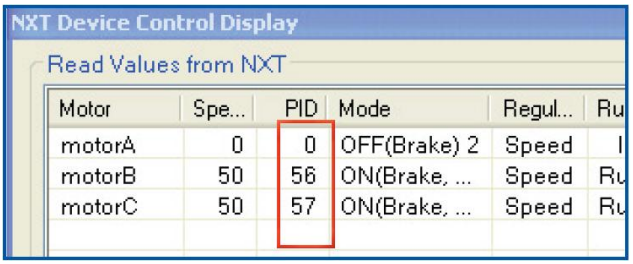
Rys. 86. Jazda w górę [2]



Rys. 87. Jazda w dół [2]

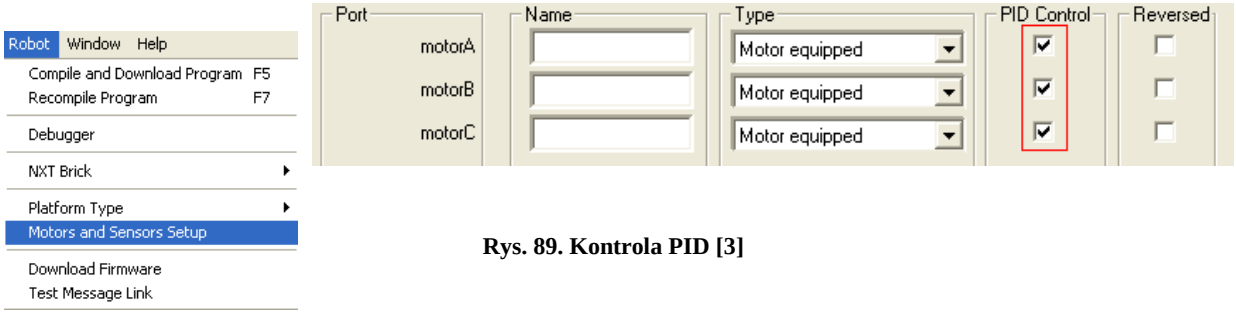
W obu przypadkach dzięki systemowi kontroli PID robot będzie poruszał się z taką samą prędkością. System ten oblicza ilość obrotów serwomotora i w razie zwiększenia bądź zmniejszenia obrotów zmienia wartość mocy aby przywrócić mu odpowiednią prędkość.

Wyniki działania kontroli PID możemy zauważyć po wrzuceniu programu na kostkę NXT i uruchomieniu jej (F5). W oknie NXT Device Control Display.



Rys. 88. PID [3]

Kontrola PID jest automatycznie włączona w każdym projekcie. Znajduje się w Robot > Motors and Sensors Setup > Motors (zakładka).



Rys. 89. Kontrola PID [3]

Jeżeli chcesz aby robot poruszał serwomotorami bez względu na powierzchnie możesz wyłączyć kontrolę PID.

10.1.4. Synchronizacja

Kontrola PID jest bardzo przydatna, jeżeli mamy zamiar używać robota w miejscu, którego podłoża nie jesteśmy pewni. Dzięki temu robot zawsze będzie zachowywać się tak jak należy. Niestety PID ma także swoje wady, jedną z nich jest to, że każdy serwomotor ma obliczanie PID oddzielne co oznacza, że wartości mocy serwomotorów mogą być różne i w skrajnych sytuacjach robot będzie sam skręcał w wyniku działania innych mocy na serwomotory. W celu zapobiegania takim sytuacjom twórcy RobotC stworzyli funkcje do synchronizacji serwomotorów.

```
nSyncedMotors;
```

Funkcja nSyncedMotors służy do ustalenia według którego serwomotora ma być ustalana ogólna wartość PID. Tworzy 1 serwomotor typu master i drugi typu slave. Przykład użycia:

nSyncedMotors = synchBC;

Synchronizacja serwomotoru C do serwomotoru B. Jeżeli chcemy zsynchronizować serwomotor B do C należy podać wartość synchCB.

nSyncedTurnRatio;

Funkcja nSyncedTurnRatio służy do ustalenia jak ma być odwzorowana wartość serwomotora master dla serwomotora slave. Przykład użycia:

nSyncedTurnRatio = 100;

Ustawienie wartości 100 oznacza, że serwomotor slave będzie zawsze przybierać dokładnie taką samą wartość jaką ma serwomotor master. Gdy ustawimy wartość -100 serwomotor skręci.

Przykładowy program:

```
task main()

{

nSyncedMotors = synchBC;

nSyncedTurnRatio = 100;
```

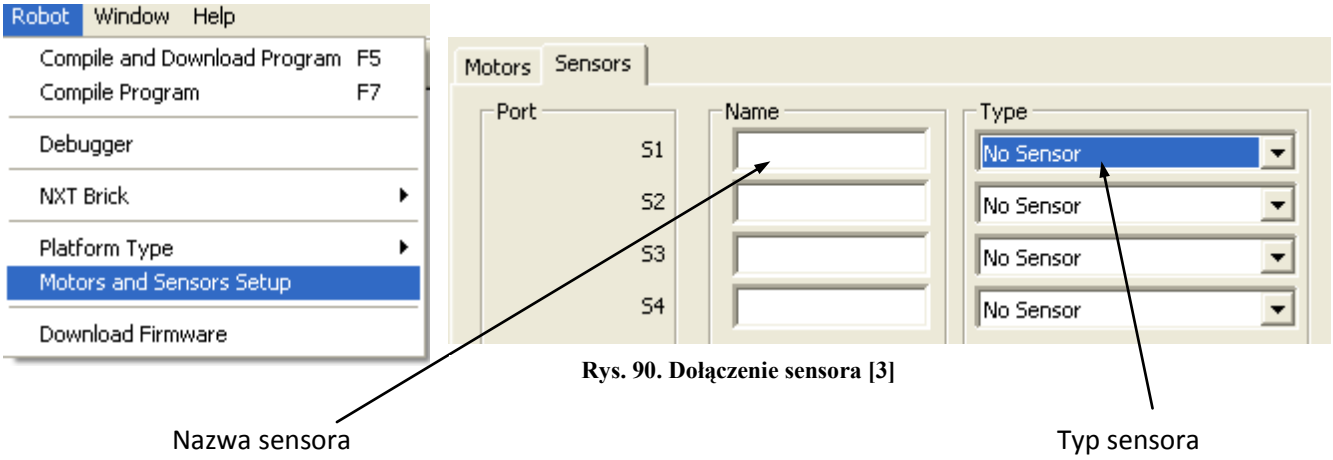
```
motor[motorB] = 50;

wait1Msec(4000);

};
```

Program ustawia serwomotor podłączony do portu B jako master i ustawia odwzorowanie 100 dla serwomotoru podłączonego do portu C. Następnie ustawia moc serwomotoru B na 50, nie trzeba ustawiać mocy serwomotoru C ponieważ odwzorowuje on moc serwomotoru B. Robot jedzie 4s do przodu po czym się zatrzymuje.

Sensory

11.Dołączanie sensora do programu

Aby móc wykorzystać sensor w programie należy na początku dołączyć go do programu. W tym celu należy wejść w Robot > Motors and Sensor Setup > Sensors (zakładka)



Rys. 90. Dołączenie sensora [3]

11.1. Typy sensorów

Touch – sensor dotyku

Light Active – sensor koloru

SoundDB – sensor dźwięku

Sonar – sensor fal ultradźwiękowych

Po dołączeniu sensora do programu wygeneruje się nam kod

#pragma config(Sensor, S1, SonarSensor, sensorSONAR)

oznaczający dołączenie sensora na 1 porcie o nazwie SonarSensor i typie sensorSONAR (Sonar).

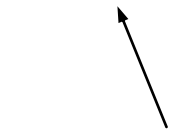
12.Programowanie sensorów

Sensory służą do odbierania bodźców ze świata zewnętrznego. Jedną z najważniejszych funkcji związanych z sensorami jest SensorValue().

```
SensorValue();
```

Funkcja służy do sprawdzania i edytowania danych wysyłanych od sensora do kostki NXT. Jest często wykorzystywana do jako warunek wykonywania pętli czy warunek w instrukcji warunkowej. Składnia:

SensorValue(LightSensor) = 30;



Nazwa sensora z którego ma być pobierana wartość

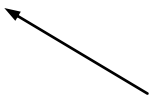
Przedstawiona jest próba przypisana wartości 30 do sensora koloru.

Operacje na sensorze dźwięku wymagają oddzielnych funkcji. Są nimi PlaySound(), ClearSounds(), MuteSound(),UnmuteSound().

```
PlaySound();
```

Funkcja służy do wydawania wbudowanych dźwięków przez robota.

PlaySound(soundBeepBeep);



Nazwa dźwięku, który ma wydawać robot

Podstawowe dźwięki robota to:

- soundBeepBeep,
- soundBlip,
- soundDownwardTones,
- soundException,
- soundFastUpwardTones,

- soundLast,

- soundLowBuzz,

- soundLowBuzzShort,

- soundShortBlip,

- soundUpwardTones.

```
ClearSounds();
```

Funkcja służy do wyłączenia dźwięku wydawanego przez robota.

```
MuteSound();
```

Funkcja służy do zatrzymania dźwięku wydawanego przez robota.

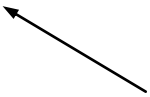
```
UnmuteSound();
```

Funkcja służy do odwieszenia dźwięku wydawanego przez robota.

RobotC ma także stałą, które służą do regulowania poziomu dźwięku. Jest to nVolume.

Składnia:

nVolume = 1;



Podana stała może przyjmować wartości z zakresu 1-4

12.1. Ćwiczenia praktyczne - sensory

Do poniższych zadań należy podłączyć do robota odpowiednie sensory w odpowiednie porty.

12.1.1. Zadanie 1.

Zaprogramuj robota aby jechał 5s z mocą taką jakie jest natężenie dźwięku dookoła niego.

Rozwiązanie:

```
#pragma config(Sensor, S1, mikro, sensorSoundDB)

task main()

{

motor[motorC] = SensorValue(mikro);

motor[motorB] = SensorValue(mikro);

Wait1Msec(5000);

}
```

Używa tutaj funkcja zwraca wartość natężenia głosu jaką pobiera sensor dźwięku i zależnie od natężenia dźwięku jedzie szybciej lub wolniej.

12.1.2. Zadanie 2.

Zaprogramuj robota aby jechał dopóki nie natknie się na przeszkodę (przy użyciu sensora dotyku). Po czym ma jechać do tyłu z mocą 75% przez 1s.

Rozwiązanie:

```
#pragma config(Sensor, S1, touchSensor, sensorTouch)

task main()

{

while(SensorValue(touchSensor) == 0)

{

motor[motorA] = 100;

motor[motorB] = 100;

}

motor[motorA] = -75;

motor[motorB] = -75;

wait1Msec(1000);

}
```

Zaprezentowane jest tutaj inne użycie funkcji SensorValue() jako warunek wykonywania pętli. Jeżeli wartość na sensorze dotyku jest równa zero, czyli jeżeli sensor dotyku nie napotkał żadnej przeszkody pętla ma się wykonywać, więc robot ma jechać dalej do przodu. W przypadku napotkania przeszkody sensor odda do programu wartość 1 i pętla się zakończy, a robot będzie wykonywać dalsze instrukcje programu, czyli powrót do tyłu o 1s.

12.1.3. Zadanie 3.

Zaprogramuj robota aby jechał do przodu dopóki jego sensor dotyku jest naciśnięty, po jego zwolnieniu ma się cofnąć mocą 75% przez 1s.

Rozwiązanie:

```
#pragma config(Sensor, S1, touchSensor, sensorTouch)
task main()
{
while(true)
{
while(SensorValue(touchSensor) == 1)
{
motor[motorA] = 100;
motor[motorB] = 100;
}
motor[motorA] = -75;
motor[motorB] = -75;
wait1Msec(1000);
}
}
```

Program jest bardzo podobny do tego z zadania wcześniejszego tylko tutaj pokazane jest jak można użyć naciśniętego sensora dotyku. Jest to przydatna operacja do np. zaprogramowania robota aby jeździł po stole z ustawionym sensorem dotyku tak aby był

przyciskany przez blat stołu (częścią mechaniczną w dół) przy czym, jeżeli sensor zostanie zwolniony, a robot dojedzie do krawędzi stołu czy innej przestrzeni robot może cofnąć się aby nie spać i wykonać inne operacje np. obrót, w tym wypadku cofa się o 1s jak w poprzednim przykładzie.

12.1.4. Zadanie 4.

Zaprogramuj robota tak aby jechał do przodu, aż napotka inny obiekt oddalony od siebie na min. 25cm (Sonar sensor) przy czym, jeżeli napotka inny obiekt ma za zadanie pojechać do tyłu z mocą 50% o 2s.

Rozwiązanie:

```
#pragma config(Sensor, S1, SonarSensor, sensorSONAR)

task main()

{

while(SensorValue(SonarSensor) > 25){

motor[motorC] = 50;

motor[motorB] = 50;

}

motor[motorC] = -50;

motor[motorB] = -50;

wait1Msec(2000);

}
```

Program działa na tej samej zasadzie jak przy użyciu sensora dotyku lecz tutaj warunkiem pętli (nadal true lub false) jest wyrażenie SensorValue(SonarSensor) > 25 , czyli czy robot znajduje się w odległości mniejszej niż 25cm od obiektu. Oczywiście dalszy ciąg zdarzeń nie powinien stanowić dla czytelnika większych problemów, dlatego tym razem omijam tłumaczenie tej części programu.

12.1.5. Zadanie 5.

Zaprogramuj robota aby wydał 3 dźwięki po 1s każdy po czym obrócił się o 360^o.

Rozwiązanie:

```
#pragma config(Sensor, S1, sensor, sensorSoundDB)

task main(){

PlaySound(soundBeepBeep);

wait1Msec(1000);

PlaySound(soundBlip);

wait1Msec(1000);

PlaySound(soundLast);

wait1Msec(1000);

motor[motorC] = -100;

motor[motorB] = 100;

wait1Msec(3200);

}
```

Programy odtwarza dźwięk przez określony czas (1s) po czym przeskakuje do następnego, po odtworzeniu 3 dźwięków obraca się o 360^o. Jest to proste wykorzystanie sensora dźwięku w robocie.

12.1.6. Ćwiczenia do samodzielnego wykonania

1. Zaprogramuj robota aby jechał do napotkania przeszkody, po jej napotkaniu ma obrócić się o 180^o i jechać nadal prosto. Ma tak dotknąć 5 przeszkód.
2. Zaprogramuj robota aby jechał do napotkania przeszkody w odległości 25cm po jej napotkaniu ma obrócić się o 180^o i jechać nadal prosto. Ma tak dotknąć 5 przeszkód.
3. Zaprogramuj robota aby jechał do przodu do napotkania przeszkody w odległości 25cm po czym ma ominąć przeszkodę (objechać dookoła, jeżeli to możliwe). Ma tak ominąć 1 przeszkodę.
4. Zaprogramuj ciekawą kompilację dźwięków wykonaną przez robota, całość ma mieć przynajmniej 5s.

5. Dołącz do wcześniej stworzonego numeru tanecznego (we wcześniejszych ćwiczeniach) dźwięk.

12.2. Komunikacja z kostką NXT

Podczas wykonywania programu standardowo na kostce wyświetlana jest jego nazwa i napis runing oznaczający, że program jest aktualnie wykonywany. Lecz można w czasie programu wykorzystywać ekran kostki do komunikacji bądź wyświetlania informacji dla użytkownika. Służą do tego funkcje:

```
eraseDisplay();
```

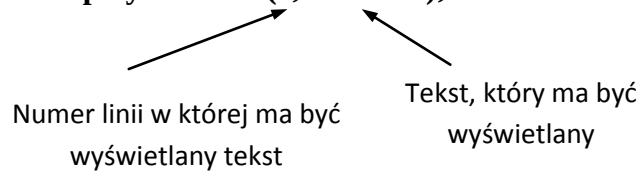
Jest to funkcja czyszcząca dotychczasową zawartość ekranu.

eraseDisplay();

```
nxtDisplayTextLine();
```

Jest to funkcja wyświetlająca tekst napis na ekranie.

nxtDisplayTextLine(0, "TEST");



Jeżeli chcemy aby tekst był wyśrodkowany należy dodać po Display słowo Centered. Komenda będzie wyglądać następująco nxtDisplayCenteredTextLine(). Natomiast jeżeli chcemy aby tekst był większy niż normalny należdy dodać przed Text słowo Big. Komenda będzie wyglądać następująco nxtDisplayBigTextLine().

Dobrym sposobem komunikacji robota z użytkownikiem jest używanie przycisków na robocie do potwierdzania / negowania decyzji itp. Twórcy udostępnili stałe odpowiadające za wartości przycisków (naciśnięty, puszczony, itp.). Są to:

```
nNxtButtonPressed
```

Stała odpowiadająca pomarańczowemu przyciskowi na kostce NXT. Przyjmuje wartości -1 gdy nie jest naciśnięty i 0-3 gdy jest naciśnięty.

```
nNxtButtonTask
```

Stała odpowiadająca za dwa przyciski do przewijania (trójkąty). Przyjmuje wartości -1 gdy nie jest naciśnięty żaden z nich i 0-9 gdy jest, któryś z nich naciśnięty.

```
nNxtExitClicks
```

Stała odpowiadająca za przycisk kończenia programu. Zwraca ile razy został naciśnięty przycisk przed wyłączeniem programu.

12.2.1. Ćwiczenia praktyczne - komunikacja z kostką NXT

12.2.1.1. Zadanie 1.

Zaprogramuj robota tak aby wyświetlał twoje imię w jednej linii, a nazwisko w drugiej przez 2s.

Rozwiązywanie:

```
task main()

{

eraseDisplay();

nxtDisplayTextLine(0,"Imie");

nxtDisplayTextLine(1,"Nazwisko");

wait1Msec(2000);

}
```

Program wyświetla przez 2s napis Imie w pierwszej linii i Nazwisko w drugiej linii.

12.2.1.2. Zadanie 2.

Zaprogramuj robota tak aby wyświetlał dzisiejszą datę przez 2s w pierwszej linii jak Big i Center Text.

Rozwiązanie:

```
task main()

{

eraseDisplay();

nxtDisplayCenteredBigTextLine(0,"24-03-2010");

wait1Msec(2000);

}
```

Progam działa jak tak jak ten z zadania 1, lecz zostało dodane do komendy pogrubienie (Big) i wyśrodkowanie (Centered).

12.2.1.3. Zadanie 3.

Zaprogramuj robota tak aby wyświetlał napis program aż nie zostanie naciśnięty przycisk NXT (pomarańczowy), po czym ma wypisać tekst zegnaj i się wyłączyć.

Rozwiązanie:

```
task main()

{

eraseDisplay();

nxtDisplayTextLine(0,"Program");

while(nNxtButtonPressed == -1){};

eraseDisplay();

nxtDisplayTextLine(0,"Zegnaj");

wait1Msec(1000);

}
```

Program standardowo wyświetla tekst, dochodzi do pętli while i ma czekać, aż wartość przycisku będzie inna niż -1 (czyli, aż przycisk nie zostanie naciśnięty) następnie wypisuje kolejny tekst i kończy pracę.

12.2.1.4. Zadanie 4.

Zaprogramuj robota tak aby wypisał tekst Naciśnij przycisk i czekał na jego naciśnięcie, po czym ma ruszyć do przodu o 1s.

Rozwiązanie:

```
task main()

{

eraseDisplay();

nxtDisplayTextLine(0,"Nacisnij przycisk");

while(nNxtButtonPressed == -1){};
```

```
motor[motorB] = 100;

motor[motorC] = 100;

wait1Msec(1000);

}
```

Program jest podobny do wcześniejszych, czyli wypisuje tekst, czeka na akcje po czym wykonuje ruch do przodu.

12.2.1.5. Zadanie 5.

Zaprogramuj robota tak aby wypisał tekst Czy Ci cos zagrac? W pierwszej linii i Naciśnij by odsłuchać w 2 linii. Jeżeli użytkownik naciśnie przycisk NXT robot ma zagrać melodyjkę przez 1s i się wyłączyć.

Rozwiązanie:

```
task main()

{

eraseDisplay();

nxtDisplayTextLine(0,"Czy Ci cos zagrac?");

nxtDisplayTextLine(1,"Nacisnij przycisk");

nxtDisplaytextLine(2,"aby odsluchac");

while(nNxtButtonPressed == -1){};

PlaySound(soundLowBuzz);

wait1Msec(1000);

}
```

Program wypisuje tekst lecz jest tutaj pewien problem, tekst w drugiej linii się nie mieści i jest ucinany, więc trzeba resztę tekstu wrzucić samodzielnie do linii trzeciej. Reszta powinna być już znana dla czytelnika.

12.2.2. Ćwiczenia do samodzielnego wykonania

1. Zaprogramuj robota aby wyświetlał napis Dzień dobry na ekranie kostki.
2. Zaprogramuj robota aby wyświetlał wyśrodkowany napis Program zrobiony przez <twoje imie>, pamiętaj, że tak długi tekst nie zmieści się w 1 linii.
3. Zaprogramuj robota tak aby jechał do przodu 1s dopiero po naciśnięciu przycisku NXT.
4. Zaprogramuj robota tak aby po naciśnięciu przycisku NXT świecił przez 1s każdym z kolorów (czerwony, niebieski, zielony) po kolei.
5. Zaprogramuj robota tak aby utwór taneczny z dźwiękiem, który stworzyłeś we wcześniejszych ćwiczeniach był uruchamiany za pomocą przycisku NXT i aby użytkownik był informowany co aktualnie robot próbuje wykonać przez ekran kostki NXT.

12.2.3. Pytania sprawdzające wiedzę - RobotC

1. Aby robot użył serwomotoru podłączonego do portu A należy użyć funkcje:
    a. motor[motorA] = 100;
    b. wait1Msec(1000);
    c. eraseDisplay();
2. Kontrola PID służy do:
    a. Pobierania wartości sensorów
    b. Regulowania mocy serwomotorów
    c. Sterowania robotem zdalnie
3. Aby robot obrócił się należy:
    a. Ustawić takie same wartości mocy serwomotorów
    b. Ustawić zerowe wartości mocy serwomotorów
    c. Ustawić przeciwne wartości mocy serwomotorów
4. Do odczytywania wartości sensorów służy funkcja:
    a. `PlaySound(soundBeepBeep);`
    b. `sensorValue(touchsensor);`
    c. `MuteSound();`
5. Do wyświetlania tekstu na ekranie kostki służy funkcja:
    a. `eraseDisplay();`
    b. `nxtDisplayTextLine(0,”tekst”);`
    c. `nNxtButtonPressed`

Podsumowanie

Właśnie dotarłeś do ostatniego rozdziału tego poradnika, staraliśmy się przekazać Ci największą ilość wiedzy na temat programowania i budowy robota Lego Mindstorms. Niestety nie sposób wyczerpać tematykę robotów w jednym poradniku. Mamy jednak nadzieję, że rozbudziliśmy twoją wyobraźnię i niedługo sam staniesz się autorem wielu ciekawych modyfikacji robota zarówno w budowie jak i w działaniu. Tutaj rozpoczyna się twoja własna przygoda z Lego Mindstorms i od Ciebie będzie zależeć jak wykorzystasz wiedzę przekazną w miniejszym poradniku.

W związku z czym życzymy Ci wielu udanych programów i modyfikacji robota!

Bibliografia

1. http://www.robonet.pl/?shop&c=lego
2. http://www.education.rec.ri.cmu.edu/previews/robot_c_products/teaching_rc_lego_v2_preview/\
3. Zrzuty ekranów z programów: RobotC i Lego Mindstorms Software

Autorzy:

1. ……..……..
2. ……..……..

Pod kierunkiem:

mgr inż. ……..……..