POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA DYPLOMOWA INŻYNIERSKA
na kierunku INFORMATYKA
specjalność: Inżynieria oprogramowania

Jakub MATYSIAK

Nr albumu: 207439

Rok akad.: 2009/2010
Warszawa, 18.11.2009

ZDALNE STEROWANIE POJAZDEM LEGO NXT

Zakres pracy:

1. *Wprowadzenie i sformułowanie celu pracy*
2. *Budowa pojazdu*
3. *Analiza gotowych rozwiązań do zdalnego sterowania kostką NXT*
4. *Opis własnego rozwiązania, implementacja i testy*
5. *Podsumowanie i wnioski*

Kierujący pracą: dr inż. Witold Czajewski

Konsultant:

Termin złożenia pracy: 25.01.2010

REMOTE CONTROL OF LEGO NXT VEHICLE

Abstract

As part of this BSc Thesis a truck capable of being remotely controlled was built exclusively out of Lego bricks . The truck was built in accordance with the rules of Lego Truck Trial competition in which Author intends to participate.

An analysis of existing solutions for remote control of the NXT brick was made. It turned out that their capabilities are not satisfactory. After that, a various programming languages and development environments which can be used to program NXT was compared. One of them – LeJOS was chosen as it proved to be the best choice for Author.

Two applications in Java language was written: the first one intended to mobile device in accordance with the Java Mobile Edition specification, the other one intended to NXT in LeJOS technology. These applications are linked together via a Bluetooth connection.

The obtained remote control model for vehicle is effective and easy-to-use. Connection between devices is fast, stable and has a wide coverage.

Spis treści

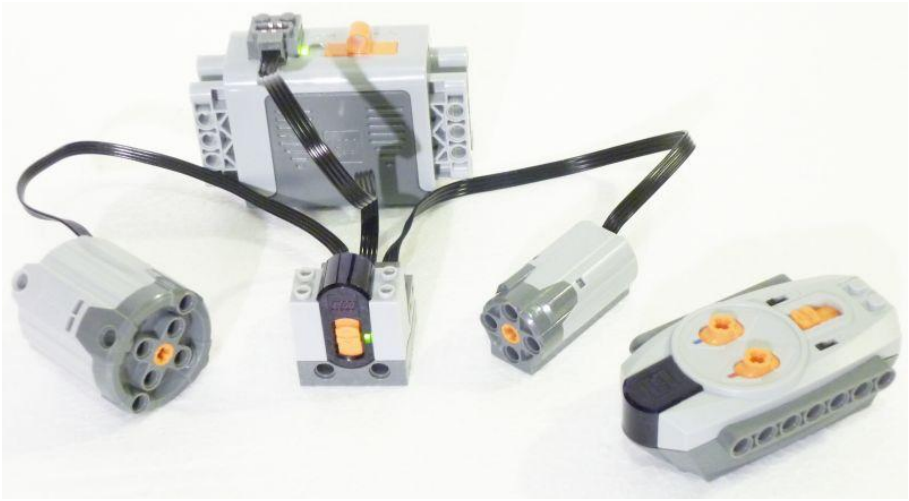
1 Wprowadzenie

1.1 Geneza pracy

W czasie mojego dzieciństwa bardzo lubiłem bawić się klockami LEGO. Po latach hobby to wróciło do mnie za sprawą pojawienia się na rynku zestawu Lego Mindstorms NXT [1] zawierającego programowalną kostkę. W tym czasie, zacząłem także uczestniczyć w zawodach Grand Prix Mazowsza [2]. Zawody te polegają na ściganiu się terenowymi pojazdami zbudowanymi we własnym zakresie wyłącznie z klocków Lego (rysunek 1, niżej).

Rysunek 1. Ciężarówka podczas pokonywania trasy rajdu.

Popularnym sposobem sterowania pojazdów jest system Lego Power Functions [3], w którego skład wchodzi m.in. pilot (numer elementu 8885) oraz odbiornik na podczerwień (numer elementu 8884). Zaletą systemu Lego Power Functions jest bardzo prosta obsługa. Odbiornik podłącza się do źródła zasilania, natomiast do odbiornika podłącza się silniki (rysunek 2, niżej).



Rysunek 2. Najważniejsze elementy systemu Lego Power Functions.

Zarówno pilot jak i odbiornik może pracować na jednej z czterech częstotliwości. Każda częstotliwość ma 2 kanały, istnieje więc możliwość niezależnego sterowania nawet do 8 silników. Aby uzyskać taką możliwość należy posiadać 4 odbiorniki (każdy odbiornik nastawiony na inną częstotliwość) oraz co najmniej jeden pilot (lub 4 aby sterować 8 silnikami jednocześnie, jednak wtedy obserwuje się opóźnienia w docieraniu sygnału).

Niestety zasięg pilota jest niewielki. W zamkniętym pomieszczeniu sterowanie nie sprawia problemów dzięki temu, że wiązka podczerwieni odbija się od ścian. W takich warunkach zasięg pilota wynosi do 10 metrów. Jednak w otwartej przestrzeni jest znacznie gorzej i aby sygnał był odbierany, pilota należy nakierowywać dokładnie na odbiornik. Dodatkowo w słonecznie dni wiązka emitowana przez pilota jest tłumiona przez światło słoneczne i zasięg może spaść nawet do 20 cm.

Możliwości regulacyjne silników w systemie Power Functions także pozostawiają wiele do życzenia. Jedyne instrukcje, jakie można wysłać to obrót danego silnika w lewo/prawo oraz „przestań obracać”. Nowy pilot (numer elementu 8879) widoczny na rysunku 3 (niżej) posiada 7 - stopniową regulację prędkości obrotów silnika, jednak funkcja ta nie ma zastosowania w rajdach terenowych.

Rysunek 3. Nowy pilot Lego PF z regulacją prędkości.

Postanowiłem poszukać jakiegoś lepszego rozwiązania do zdalnego sterowania. W związku z tym, że według regulaminu [2] w pojeździe można wykorzystać jedynie oryginalne elementy Lego, wybór nie był trudny. Zdecydowałem się wykorzystać kostkę NXT i sterować ciężarówką za pomocą urządzenia mobilnego poprzez połączenie Bluetooth. Mając już doświadczenie z zestawem NXT zdobyte podczas projektu indywidualnego na 5 semestrze wiedziałem, że możliwości kostki w takim zastosowaniu są duże.

Jedyne urządzenie mobilne, jakie posiadam to telefon komórkowy Nokia E51, która obsługuje aplikacje Java. Po wypróbowaniu istniejących programów działających na moim

telefonie stwierdziłem, że warto samemu napisać aplikację dokładnie dostosowaną do moich potrzeb.

Zależało mi na niezawodnej komunikacji o dużym zasięgu. Oprócz tego chciałem precyzyjnie sterować układem skrętnym oraz przede wszystkim skrzynią biegów, co bez NXT jest trudne do osiągnięcia.

1.2 Cel pracy

Celem pracy jest budowa pojazdu o dużych możliwościach terenowych, zgodnego z regulaminem zawodów [2] oraz napisanie aplikacji na urządzenie mobilne i na kostkę NXT, które pozwolą na pełną kontrolę nad pojazdem.

Aplikacja na urządzenie mobilne ma uruchamiać się i działać na telefonach komórkowych Nokia obsługujących aplikacje Java, w szczególności na telefonie Nokia E51.

Dzięki aplikacjom tym możliwe ma być zdalne sterowanie zbudowanym pojazdem. Aplikacje mają mieć łączność między sobą poprzez połączenie Bluetooth. Ustanawianie połączenia pomiędzy urządzeniami ma być możliwie szybkie, a samo połączenie stabilne. Program ma być prosty i wygodny w użytkowaniu oraz odporny na zdarzenia zewnętrzne, np. zablokowanie silników.

2 Budowa pojazdu

2.1 Wymagania techniczne

Aby pojazd został dopuszczony do zawodów musi spełnić kilka warunków. Warunki te mają na celu zwiększenie realizmu oraz poziomu trudności budowy pojazdów. Poniżej znajdują się najważniejsze wymagania wymienione w regulaminie [2]:

- dopuszczalne są modele samochodów ciężarowych lub odpowiadające im pojazdy specjalne w skali ok. 1:13, model powinien być możliwie podobny do oryginału
- liczba i układ osi skrętnych muszą być zgodne z oryginałem.
- wszystkie osie muszą mieć ruchome, pracujące zawieszenie
- wszystkie koła muszą być napędzane
- faktyczny ciężar własny pojazdu musi wynosić minimum 1 kg
- zabronione jest stosowanie modyfikowanych klocków lub używanie elementów nie-Lego

W celu wyrównania szans zawodników posiadających różne silniki napędowe stosuje się przelicznik. Przelicznik ten także premiuje pojazdy o większej liczbie osi oraz o większej wadze, aby zachęcić uczestników do budowania większych ciężarówek.

$$\text{przelicznik} = \frac{\text{masa pojazdu wyrażona w gramach} * \text{wykładnik osiowy}}{\text{sumaryczna moc mechaniczna silników napędowych dla napięcia 9V}}$$

Wykładnik osiowy dla pojazdów dwuosiowych wynosi 1. Za każdą kolejną oś pojazdu jest on powiększany o 0,2.

Przelicznik dla zbudowanego przeze mnie pojazdu wynosi 2048. Jest to całkiem dobry wynik, zważywszy, że w 2009 roku średni przelicznik pojazdów startujących w jednych z zawodów wynosił 1005.

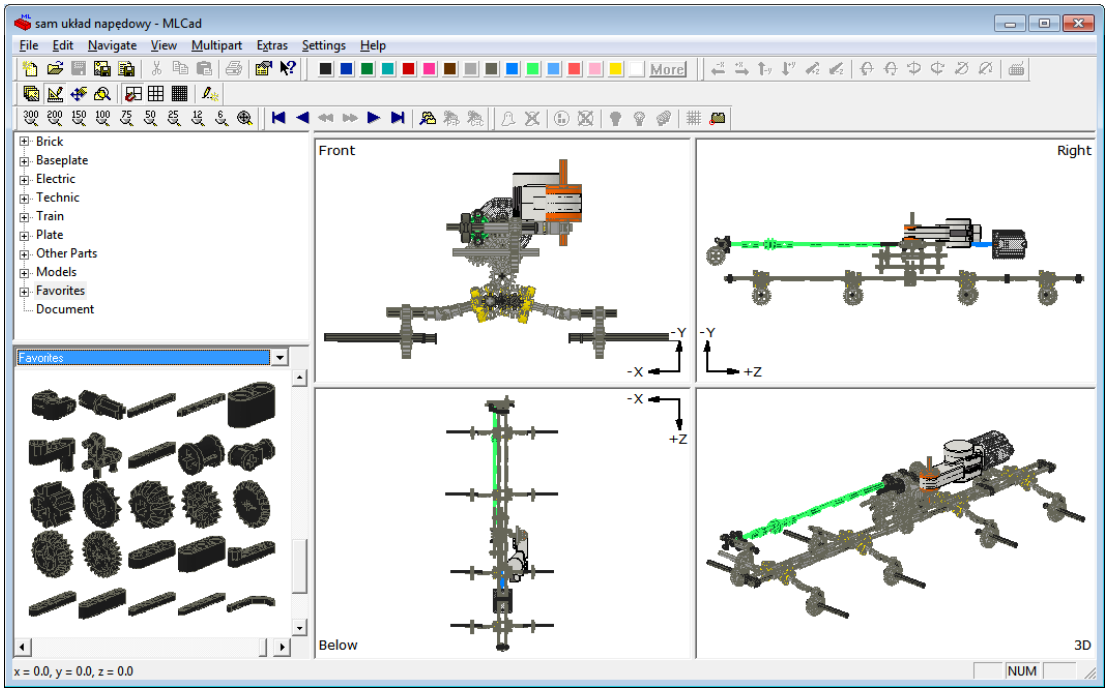
Punkty za przejazd odcinka rajdu oblicza się następująco:

$$\text{wynik} = \frac{\text{przelicznik}}{\text{czas przejazdu odcinka wyrażony w sekundach}}$$

2.2 Projektowanie na komputerze

W budowie z Lego pomocne są programy CAD, dzięki którym możliwe jest zaprojektowanie konstrukcji przed składaniem jej z rzeczywistych klocków. Co prawda rzadko się zdarza, że całość modelowana jest najpierw w programie, chyba, że „budowniczy" nie posiada odpowiedniej liczby rzeczywistych klocków. W przypadku skomplikowanych modeli warto jest sprawdzić, czy np. dane połączenie klocków jest w rzeczywistości możliwe bez generowania naprężeń.

Programy CAD wykorzystuje się także do dokumentacji już wykonanych budowli. Tak właśnie uczyniłem w przypadku zbudowanej przeze mnie ciężarówki. Występujące w późniejszej części tej pracy rysunki z wyrenderowanymi elementami samochodu zostały stworzone za pomocą programu MLCad (rysunek 4, niżej), a następnie wyrenderowane programem LDView. Obie te aplikacje wchodzą w skład pakietu narzędzi LDraw [4].



Rysunek 4. Okno programu MLCad.

2.3 Analiza możliwych rozwiązań

Rozwiązania tutaj prezentowane mogą być, z uwagi na ograniczenia klocków Lego, pewnymi uproszczonymi modelami układów stosowanych w rzeczywistych samochodach.

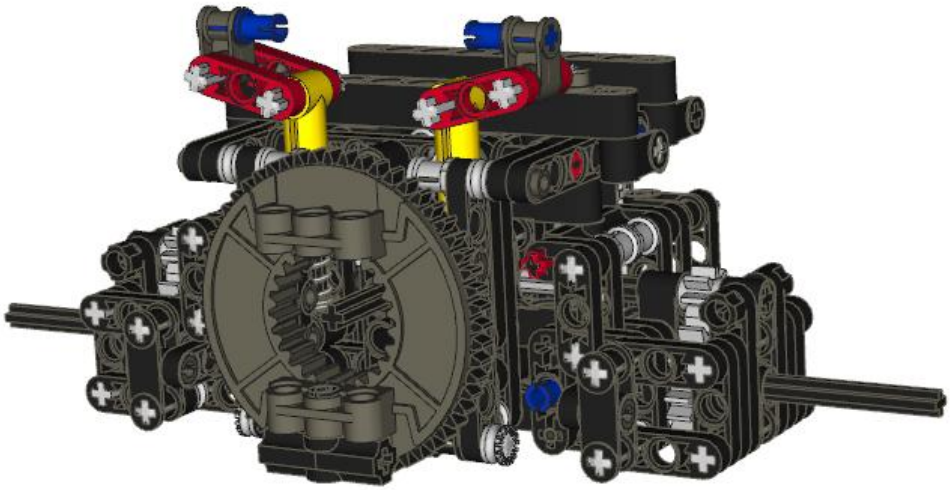
2.3.1 Układ podwozia

Jako układ podwozia mam na myśli liczbę osi oraz możliwość skrętu kół na tych osiach. Wyznacznikami dobrego układu podwozia do rajdów Lego Truck Trial jest mały promień skrętu oraz możliwość pokonywania dużych rowów. Popularne konfiguracje to

4x4x2, 4x4x4, 6x6x2, 6x6x4, 8x8x4 i 8x8x8. Po przeprowadzeniu kilku testów zdecydowałem się na układ 8x8x4, ponieważ większa liczba osi pozwalała pokonywać większe rowy, promień skrętu był wystarczająco mały, a brak skrętnych kół na dwóch ostatnich osiach umożliwiał ich trwalszą i prostszą konstrukcję. Nie bez znaczenia była także premia do przelicznika za wykładnik osiowy opisany w punkcie 2.1.

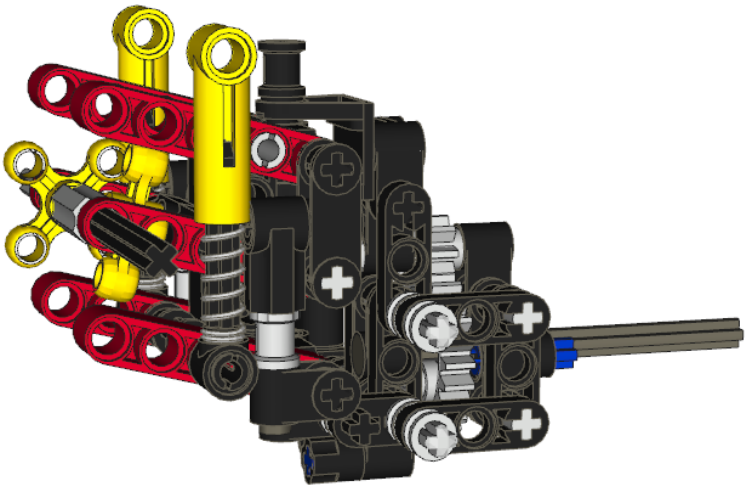
2.3.2 Zawieszenie

Mimo, że zawieszenie jest wymagane przez regulamin to jego konstrukcja odgrywa ważną rolę w możliwościach terenowych pojazdów z Lego. Pożądany jest dobry wykrzyż, ponieważ dzięki niemu możliwe jest pokonywanie dużych przeszkód. Inaczej mówiąc, ważne jest, aby podczas pokonywania przeszkody możliwie jak najwięcej kół miało stały kontakt z podłożem. W osiągnięciu tego celu dobrze sprawdzają się różnego rodzaju mosty kolebkowe (rysunek 5, niżej). Trzeba też zaznaczyć, że w przypadku kolebek, określenie zawieszenie jest nieco na wyrost, ponieważ most taki nie amortyzuje uderzeń w podłoże.



Rysunek 5. Most kolebkowy.

Oprócz wykrzyżu, drugą ważną cechą zawieszenia jest stabilność. Mosty kolebkowe niestety w kwestii stabilności, szczególnie poprzecznej, wypadają bardzo słabo. Znacznie lepsze pod tym względem jest zawieszenie niezależne. Zawieszenie niezależne oparte na pojedynczych wahaczach jest w miarę proste do zbudowania i jednocześnie stabilne, jednak z uwagi na małą masę pojazdu zdolność dostosowania się tego zawieszenia do podłoża jest stosunkowo niewielka. Dlatego w mojej ciężarówce zastosowałem zawieszenie niezależne oparte na podwójnych wahaczach (rysunek 6, niżej). Co prawda możliwy do osiągnięcia wykrzyż jest mniejszy niż w przypadku zawieszenia zależnego, ale stabilność oraz zdolność dopasowania się do podłoża jest na bardzo dobrym poziomie.



Rysunek 6. Podwójny wahacz wraz ze zwrotnicą.

Dodatkowo w osiągnięciu lepszego wykrzyżu pomaga, na pierwszy rzut oka niepożądana cecha wynikająca z miękkości plastiku, z którego wykonane są klocki - giętkość ramy. Po wielu testach okazało się jednak, że nie wpływa ona negatywnie na przeniesienie napędu, a poprawia stabilność.

2.3.3 Silniki napędowe i przeniesienie napędu

Do napędu ciężarówki wybrałem silnik Lego Power Functions XL. Można go podłączyć bezpośrednio do kostki NXT po zastosowaniu dwóch kabli – przejściówek sprzedawanych przez Lego (rysunek 7, niżej).

Rysunek 7. Po lewej silnik Power Functions XL wraz z przejściówkami, po prawej silnik NXT.

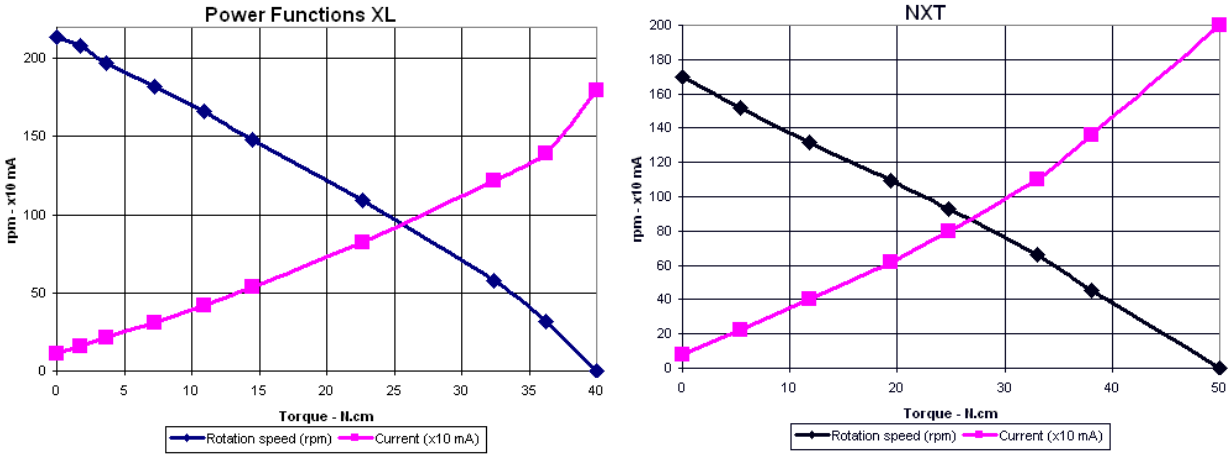
Silnik ten jest znacznie mniejszy i poręczniejszy niż oryginalny silnik z zestawu NXT. Nie posiada tachometru, ale w układzie napędowym nie jest on potrzebny. Silnik napędza oś zaznaczoną na rysunku 9 (strona 12) na niebiesko.

Poniżej znajduje się tabela zestawiająca charakterystyki obu silników przy napięciu 9V pod obciążeniem (tabela 1).

Silnik	Moment obrotowy [N.cm]	Obroty [rpm]	Prąd [A]	Moc mechaniczna [W]	Moc elektryczna [W]	Skuteczność
PF XL	14,5	146	0,55	2,21	4,95	45%
NXT	16,7	117	0,55	2.03	4,95	41%

Tabela 1. Charakterystyki silników przy napięciu 9V pod obciążeniem.

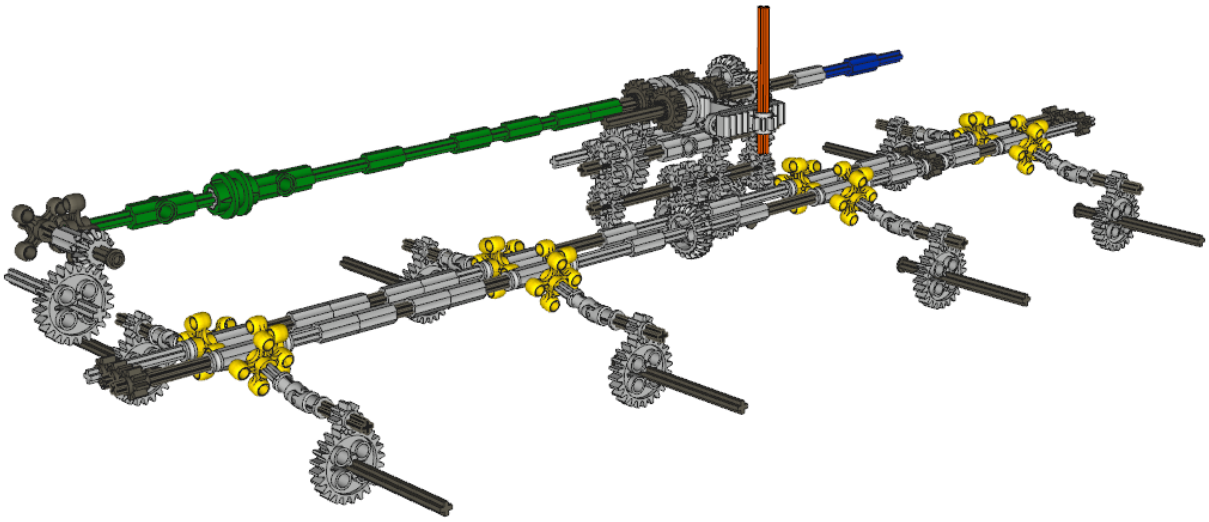
Na rysunku 8 (niżej) przedstawione są wykresy charakterystyk prędkości obrotowej od momentu obrotowego silników.



Rysunek 8. Wykresy charakterystyk silników.

Powyższe dane (tabela 1 oraz rysunek 8) pochodzą ze strony Philippe E. Hurbain [6]. Oba silniki mają wewnątrz taki sam silnik elektryczny. Jak widać silnik Power Functions XL jest nieco szybszy oraz mocniejszy. Jest to spowodowane mniejszą wewnętrzną redukcją oraz prawdopodobnie brakiem tachometru.

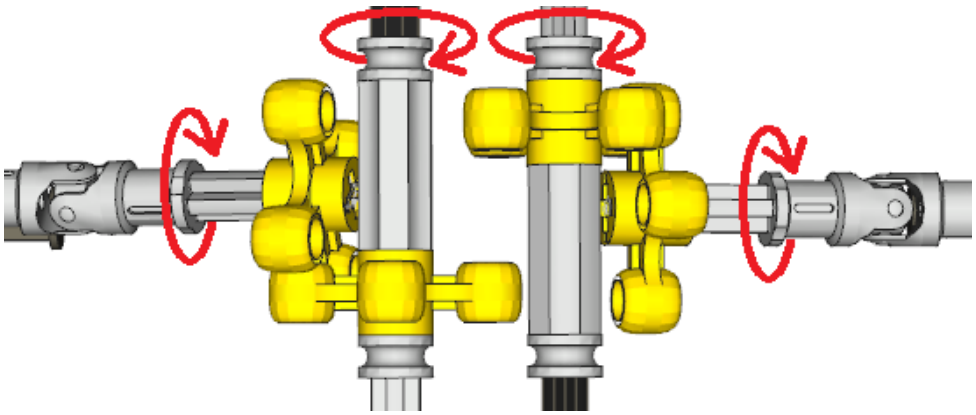
Układ przeniesienia napędu jest krytycznym elementem pojazdu przygotowywanego na zawody Lego Truck Trial, znacznie ważniejszym niż zawieszenie czy układ skrętny. Budując z klocków trzeba przyłożyć szczególną uwagę na mocne obudowanie wszelkich mechanizmów, aby zębatki nie przeskakiwały przy wysokim momencie obrotowym. Cały układ napędowy widoczny jest na rysunku 9 (niżej).



Rysunek 9. Układ napędowy ciężarówki.

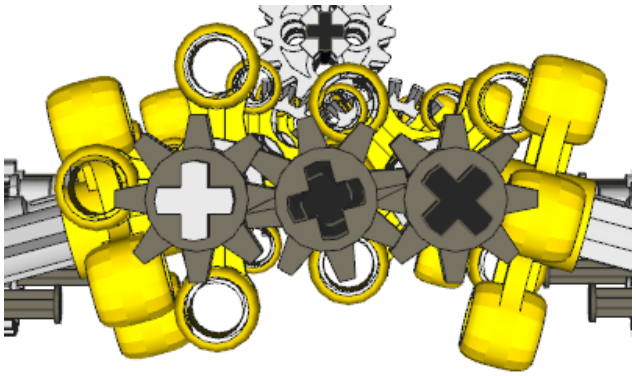
Aby uniknąć zbędnych naprężeń w skrzyni biegów, znajduje się ona zaraz za silnikiem, przed jakimikolwiek zwiększeniem przełożenia. Jest ona przystosowana do obsługi 4 przełożeń (18.75:1, 11.25:1, 6.25:1 i 3.75:1), jednak z powodu usterki sensora podczerwieni Hitechnic IRLink podczas pisania pracy, byłem zmuszony do zmniejszenia liczby zdalnie obsługiwanych przełożeń do dwóch (11.25:1 i 3.75:1). Silnik zmieniający przełożenia podłączony jest do pomarańczowej ośki (rysunek 9, wyżej).

Za skrzynią biegów napęd przenoszony jest na dwa wały główne. Dwa wały są potrzebne po to, aby lewe oraz prawe koła kręciły się w tę sama stronę. Jak widać na rysunku 10, oba wały oraz lewa i prawa półoś obracają się w tym samym kierunku.



Rysunek 10. Wały główne napędzające półosie.

Widoczne na rysunku 10 (wyżej) żółte elementy potocznie zwane knobami są jedynymi kątowymi zębatkami produkowanymi przez Lego wystarczająco mocnymi do przenoszenia napędu pod dużym obciążeniem. Niestety ich mała liczba zębów (4) powoduje efekt skokowego przekazywania napędu pod obciążeniem. Aby zminimalizować ten efekt prawy wał jest obrócony względem lewego o 45 stopni jak to widać na rysunku 11 (niżej).



Rysunek 11. Wały lewy i prawy są obrócone względem siebie o 45 stopni.

Dzięki temu lewe koła krytyczny moment skokowego przekazu napędu mają w innym czasie niż prawe koła. Dodatkowym czynnikiem mającym wpływ na ten negatywny efekt jest miękkość plastikowych osi. Ciemnoszare zębatki widoczne na rysunku 11 (wyżej) służą do dodatkowego synchronizowania wałów zmniejszając skręcanie się osi. Zębatki o takiej funkcji zamontowane są w każdej sekcji pojazdu (rysunek 18, strona 17), aby wały nie rozsynchronizowywały się przy serwisie (odłączeniu jednostki napędowej).

Na samym końcu układu przeniesienia napędu, zaraz przed każdym z kół zastosowane są zwolnice z dużym przełożeniem 3:1. Służą one zwiększeniu momentu obrotowego oraz zmniejszeniu naprężeń wewnątrz układu napędowego pojazdu.

Aby w każdym momencie mieć kontrolę nad pojazdem ważne jest, aby napęd był przekazywany stale na wszystkie koła. Dzięki temu samochód, nawet gdy niektóre koła stracą kontakt z podłożem, będzie mógł w dalszym ciągu jechać. Dlatego też ciężarówka nie posiada dyferencjałów.

2.3.4 Układ skrętny

Układ skrętny jest dość skomplikowany z powodu faktu, że skręcane są dwie pierwsze osie. Konstrukcja zwrotnic (rysunek 6, strona 10) pozwoliła na maksymalny kąt skrętu kół pierwszej osi $\alpha = 38°$. Oś druga musi skręcać się proporcjonalnie mniej niż oś pierwsza.

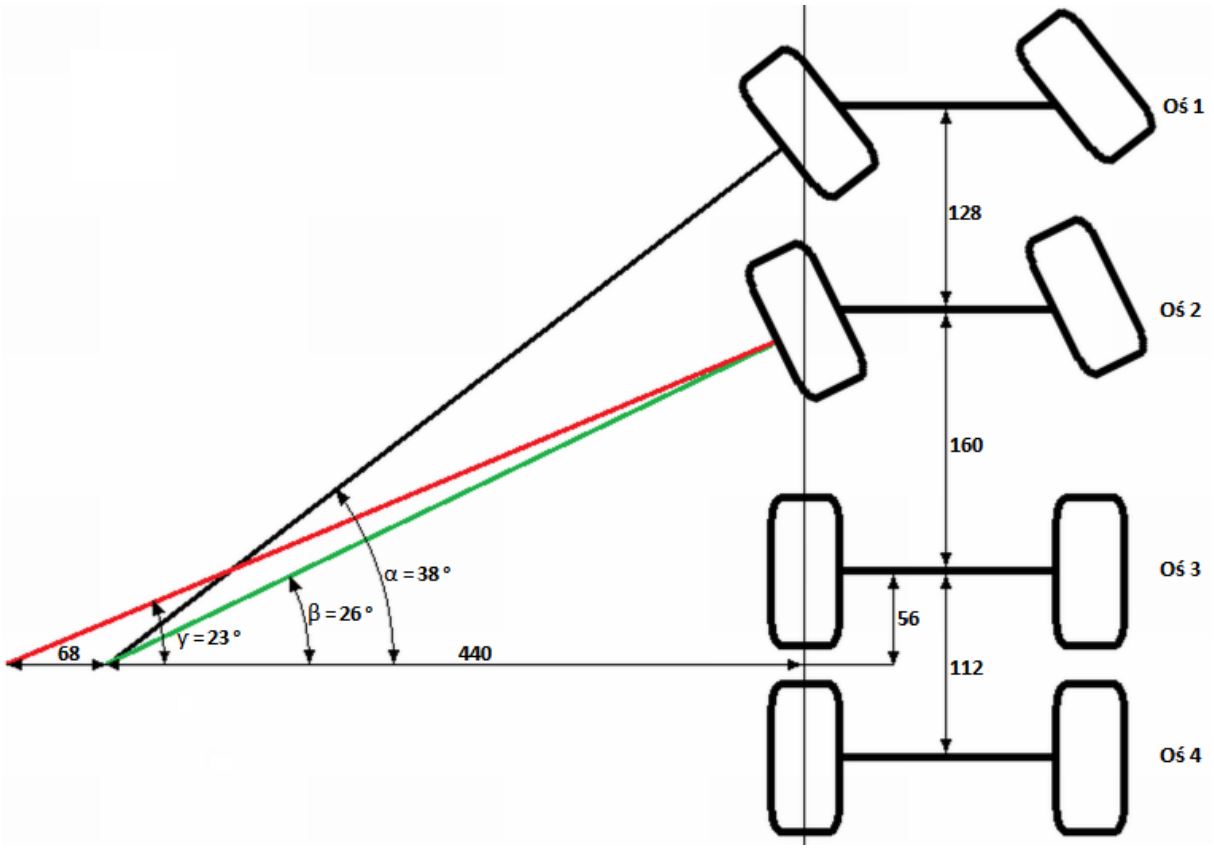
Współczynnik przełożenia napędu drążka środkowego osi 2 do drążka środkowego osi 1 wyliczyłem w następujący sposób [5].

$$promień\ skrętu = \frac{\left(128 + 160 + \frac{112}{2}\right)}{\tan(38°)} = 440$$

$$\beta = tan^{-1}\left(\frac{216}{440}\right) = 26°$$

$$przełożenie = \frac{38°}{26°} = 1{,}46$$

Niestety najbliższym, prostym do uzyskania przełożeniem w klockach Lego jest przełożenie uzyskane z zębatek z12 i z20, czyli 1,66:1. Z uwagi na brak miejsca w pojeździe na bardziej skomplikowane konfiguracje zębatek oraz ryzyko zbyt dużych luzów zastosowałem to właśnie przełożenie. Na poniższym schemacie (rysunek 12) zieloną linią zaznaczony jest wyliczony promień dla osi 2, zaś czerwoną linią zaznaczony jest rzeczywisty uzyskany promień. Jak widać, maksymalny kąt γ skrętu kół osi drugiej wynosi 23° zamiast wyliczonych 26°, a środek promienia skrętu przesunięty jest o 68 mm.

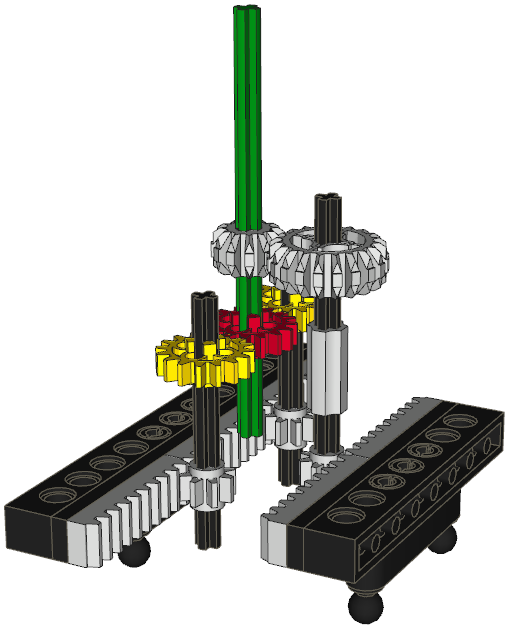


Rysunek 12. Schemat ukazujący przesunięcie promienia skrętu drugiej osi. Jednostki długości podane są w mm.

Niedokładność ta nie jest jednak znaczącym problemem, zważywszy, że samochód nie posiada dyferencjałów, geometria sterowania Ackermana nie jest zachowana a oś obrotu zwrotnic jest nieco przesunięta względem środka kół. Jednak niewielka masa pojazdu zmniejsza wagę tych kwestii. Ponadto, pojazd będzie jeździł głównie po nawierzchniach o małej przyczepności.

Sam mechanizm skrętu widoczny jest na rysunku 13 (niżej). Silnik napędowy podłączony jest do zielonej ośki. Dwie szare zębatki na samej górze odpowiedzialne są za zmianę przełożenia dla drugiej osi. Zielona oś nie może bezpośrednio poruszać środkowym drążkiem, pierwszej osi ponieważ wtedy osie skręcałyby w różnych kierunkach. W związku z tym czerwona zębatka napędza dwie żółte zębatki (z przełożeniem 1:1) odwracając kierunek

obrotu osi wyjściowych. Dodatkowo, zabieg ten zwiększa stabilność przedniego, dłuższego drążka.



Rysunek 13. Mechanizm napędzający drążki środkowe układu skrętnego.

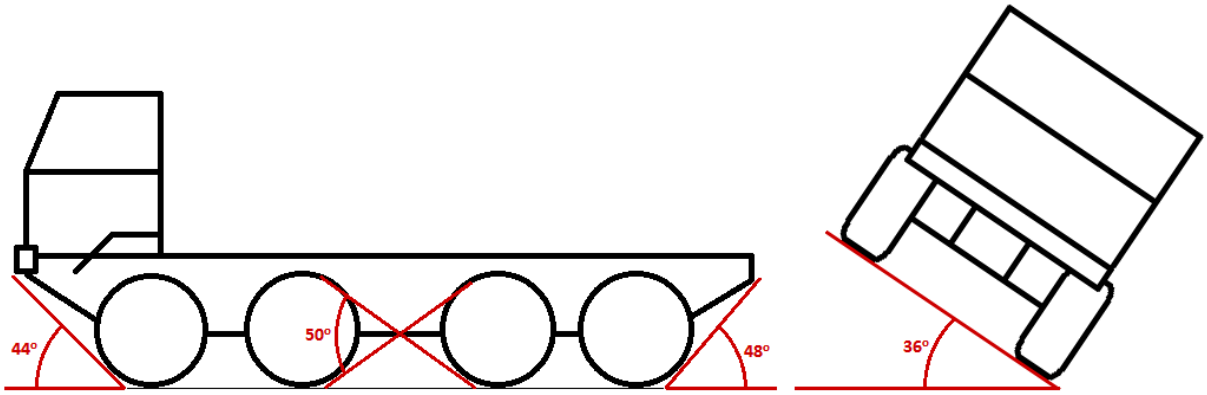
2.4 Dane techniczne i funkcje pojazdu

Pojazd to model ciężarówki Volvo FL12 8x8 wykonany w skali 1:13 (rysunek 14).

Rysunek 14. Ciężarówka gotowa do jazdy.

Napędzany jest silnikiem Lego PF XL o mocy 2,21 W. Posiada 4–biegową skrzynię biegów, jednak w momencie pisania pracy, z uwagi na ograniczoną ilość silników, wykorzystywane są jedynie 2 przełożenia. Prędkość maksymalna na 4 biegu wynosi 0,2 m/s (0,72 km/h). Masa całkowita pojazdu gotowego do jazdy równa jest 3290 g, co daje przelicznik równy 2048. Całkowita długość wynosi 620 mm, szerokość 215 mm a wysokość

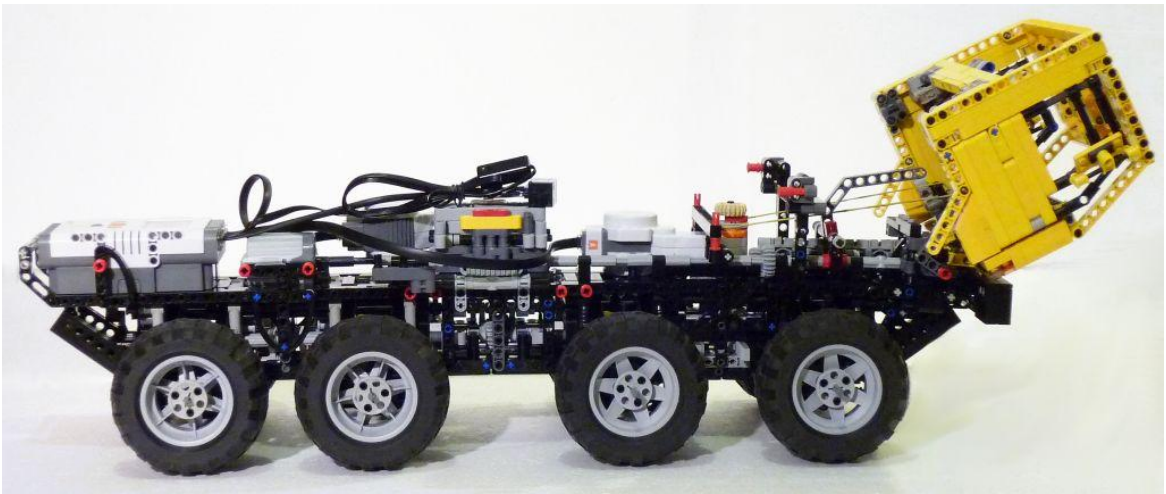
145 mm. Prześwit jest równy 41 mm. Kąt natarcia wynosi 44°, kąt zejścia 48°, kąt rampowy (grzbietowy) 50° a maksymalne przychylenie boczne to 36°. Na poniższym schemacie (rysunek 15) dane te przedstawione są w postaci graficznej.



Rysunek 15. Możliwości terenowe ciężarówki.

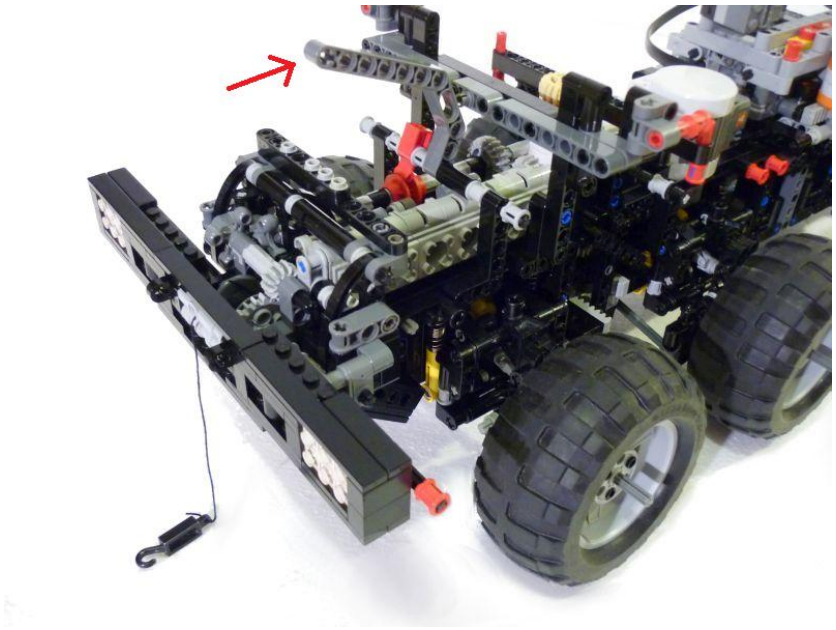
Maksymalne wzniesienie, na które zdolny jest wjechać ten samochód, jeśli tylko przyczepność będzie wystarczająca, może być nachylone do podłoża o 55°. Gdyby wykorzystać nieużywane przełożenie w skrzyni biegów, kąt ten by się jeszcze zwiększył.

Kabinę można łatwo podnieść (rysunek 16, niżej) lub całkowicie zdemontować (rysunek 17, strona 17).



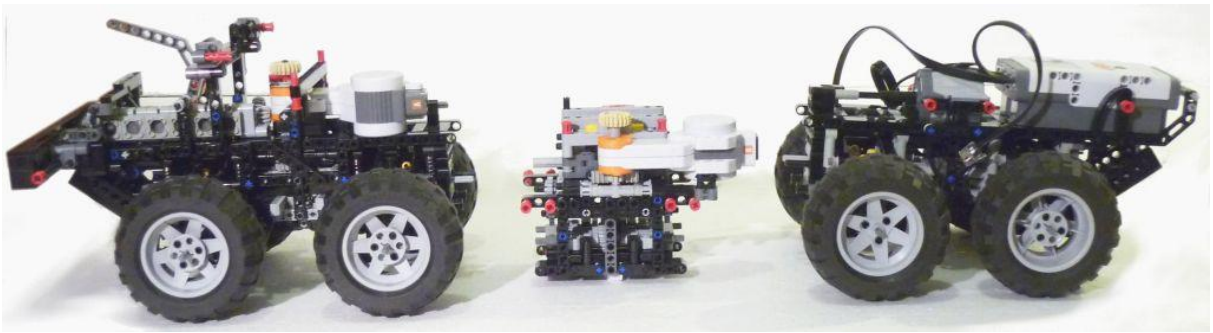
Rysunek 16. Ciężarówka z podniesiona kabiną.

Widoczna na rysunku 17 (niżej) wajcha zaznaczona strzałką służy do ręcznego załączania wyciągarki. Napęd z silnika napędowego jest przekazywany do wyciągarki z przełożeniem 3,33:1 za pomocą wału zaznaczonego na rysunku 9 (strona 12) na zielono. Wyciągarka może pracować razem z napędem kół (wrzucony jakikolwiek bieg) lub autonomicznie (wrzucony luz). Wyciągarka jest na tyle mocna, że jest w stanie podnieść cały pojazd.



Rysunek 17. Ciężarówka ze zdjętą kabiną i odczepionym hakiem wyciągarki.

Podczas zawodów może zajść potrzeba przeprowadzenia serwisu samochodu, dlatego ważne jest, aby dostęp do układów narażonych na usterki był łatwy i szybki. Zbudowany przeze mnie model można szybko (w kilkanaście sekund) rozłożyć na 3 główne sekcje (rysunek 18, niżej) – przednią część z osiami skrętnymi, środkową ze skrzynią biegów oraz tylną z dwoma ostatnimi osiami. Dzięki temu naprawa lub wymiana zepsutych elementów nie sprawia problemów.



Rysunek 18. Rozczepione podwozie pojazdu pozwalające na serwis.

3 Zestaw Lego Mindstorms NXT

3.1 Opis zestawu

Lego Mindstorms NXT to zestaw o wydany w 2006 roku przez Lego Group [1] będący następcą zestawu Mindstorms Robotics Invention System z 1998 roku. W jego skład wchodzi mikrosterownik zwany kostką, sensory, serwomotory oraz zwykłe klocki.

Istnieje kilka wersji zestawów: wersja podstawowa 8527 (rysunek 19, niżej), wersja edukacyjna 9797 z akumulatorem i nieco innymi sensorami i klockami oraz NXT 2.0 (numer 8547) z 2009 roku zawierająca nowy czujnik kolorów i inne klocki. Sama kostka w każdej wersji jest taka sama. Ja do budowy ciężarówki wykorzystałem wersję podstawową.

Rysunek 19. Podstawowa wersja zestawu Lego Mindstorms NXT o numerze 8527.

Sercem systemu jest programowalna za pomocą komputera PC lub Mac kostka NXT. Wyposażona jest w cztery porty wejścia do komunikacji z sensorami i trzy porty wyjścia sterujące serwomotorami. Kostka NXT posiada graficzny wyświetlacz LCD 100x64 pikseli, cztery przyciski służące do nawigacji po menu i programach użytkownika oraz głośnik. Ponadto wyposażona jest w kontroler Bluetooth, który umożliwia nie tylko bezprzewodowe przesyłanie programów, ale także komunikację z telefonem komórkowym, palmtopem, komputerem czy nawet inną kostką NXT. Oczywiście możliwe jest także podłączenie do komputera za pomocą kabla USB. NXT może być zasilane przez 6 baterii/akumulatorów R6 (AA) lub akumulator Li-Ion.

Poniżej znajdują się szczegółowe dane kostki NXT:

- 32-bitowy procesor 48 MHz (256 KB pamięci flash, 64 KB RAM)
- 8-bit ATmega48 mikrokontroler 4 MHz

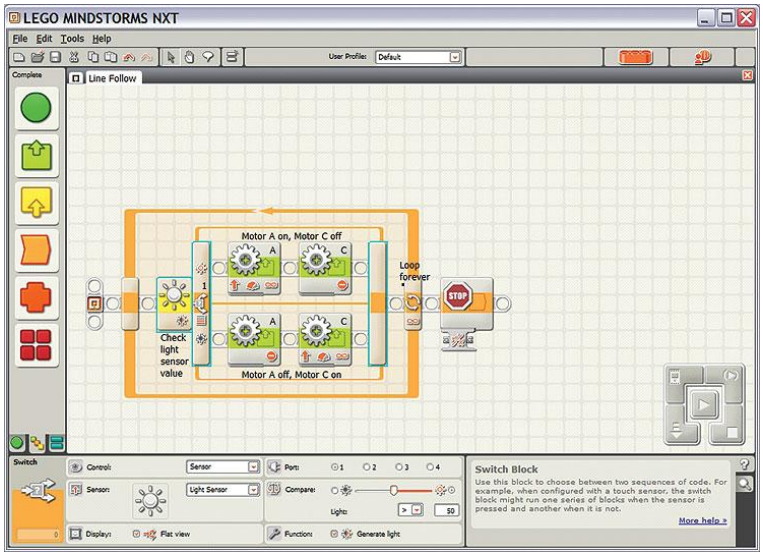
- Kontroler Bluetooth CSR BlueCore 4 @ 26 MHz (8 MBit pamięci zewnętrznej flash, 47 KB RAM)
- Port USB 1.1 (12 Mbit/s)

3.2 Popularne języki programowania

Jest wiele sposobów pisania programów wykonywalnych na kostce NXT. Istnieją rozwiązania zarówno dla dzieci jak i zaawansowane języki o dużych, jak na tak niewielkie urządzenie, możliwościach. Języki wymienione w tym rozdziale nie wyczerpują listy wszystkich dostępnych sposobów programowania. Są to jednak najpopularniejsze języki wykorzystywane do programowania kostki NXT.

3.2.1 NXT-G

NXT-G [7] to język używany w graficznym środowisku programistycznym Lego Mindstorms NXT (rysunek 20, niżej). Środowisko to jest standardowo dołączane do zestawu LEGO Mindstorms NXT 8527. Jest ono przeznaczone przede wszystkim dla dzieci i osób, które nie miały wcześniej styczności z programowaniem. Programy w tym środowisku tworzy się poprzez przeciąganie różnych bloków (instrukcji dla kostki) i ustawianie ich parametrów. Napisanie zaawansowanego programu w języku NXT-G jest trudne z uwagi na niewygodną nawigację po rozbudowanym „kodzie" oraz ograniczenia samego języka. Największą wadą środowiska jest mała szybkość programów tworzonych za jego pomocą.

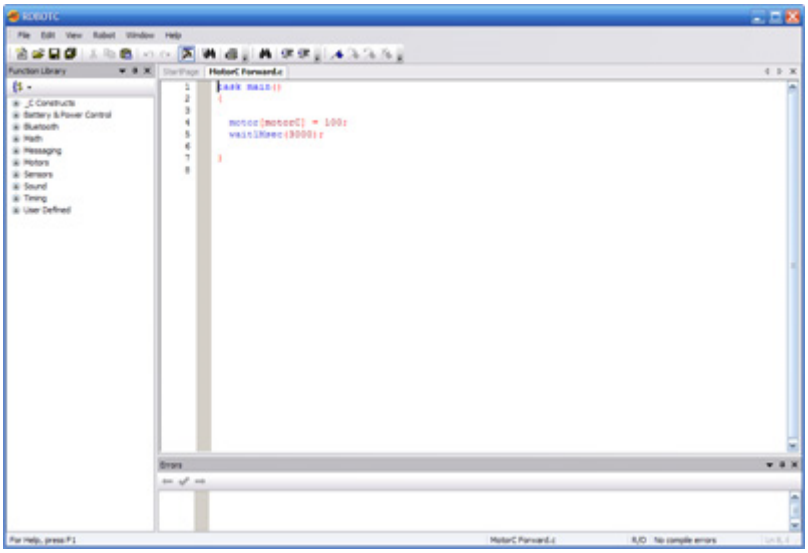


Rysunek 20. Środowisko programistyczne Lego Mindstorms NXT.

3.2.2 ROBOTC

ROBOTC [8] umożliwia pisanie programów na kostkę NXT używając języka C. Istnieje możliwość uruchamiania programów ROBOTC na kostce ze standardowym firmware Lego. W takim przypadku programy te nie mogą zawierać funkcjonalności wykraczającej poza funkcjonalność języka NXT-G. Jest jednak możliwość wgrania na kostkę firmware ROBOTC i uruchamiania bardziej zaawansowanych programów z dodatkową funkcjonalnością. Programy napisane w tym języku mogą być wykonywane nawet ponad 100 razy szybciej niż programy w NXT-G [12].

Środowisko programistyczne ROBOTC (rysunek 21, niżej) posiada, podobnie jak NXT-G funkcję przeciągania bloków z listy po lewej stronie okna, z tym, że w ROBOTC przeciąga się tekst (kod) który następnie można edytować. ROBOTC nie jest darmowe do użytku.

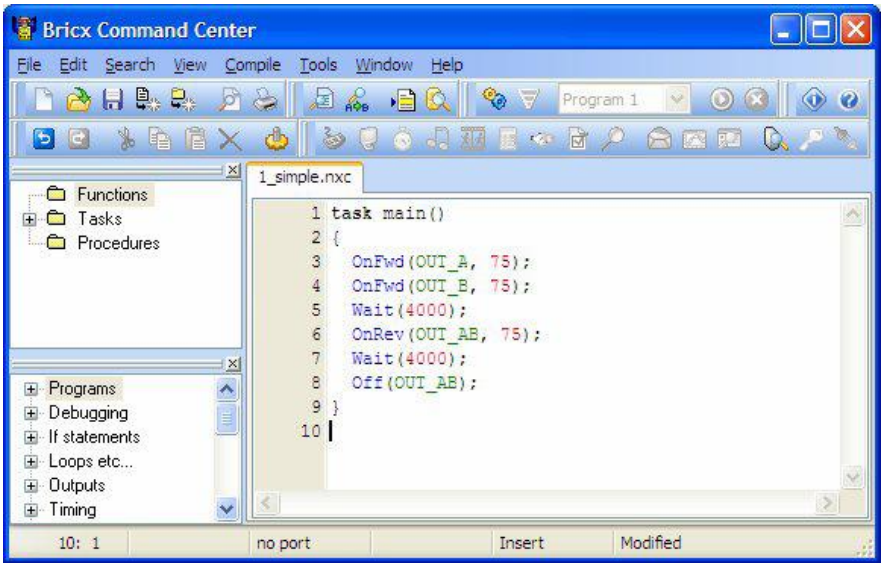


Rysunek 21. Środowisko programistyczne RobotC.

3.2.3 NXC

NXC [9] czyli Not Exactly C to język programowania podobny do C przeznaczony do programowania NXT. NXC nie wymaga instalowania niestandardowego firmware do kostki, dzięki czemu możliwe jest jednoczesne posiadanie na kostce programów w NXC jak i NXT-G. Wadą są ograniczenia języka takie same jak w przypadku NXT-G. Programy napisane w NXC są ponad dwudziestokrotnie szybciej wykonywane niż te w NXT-G [12].

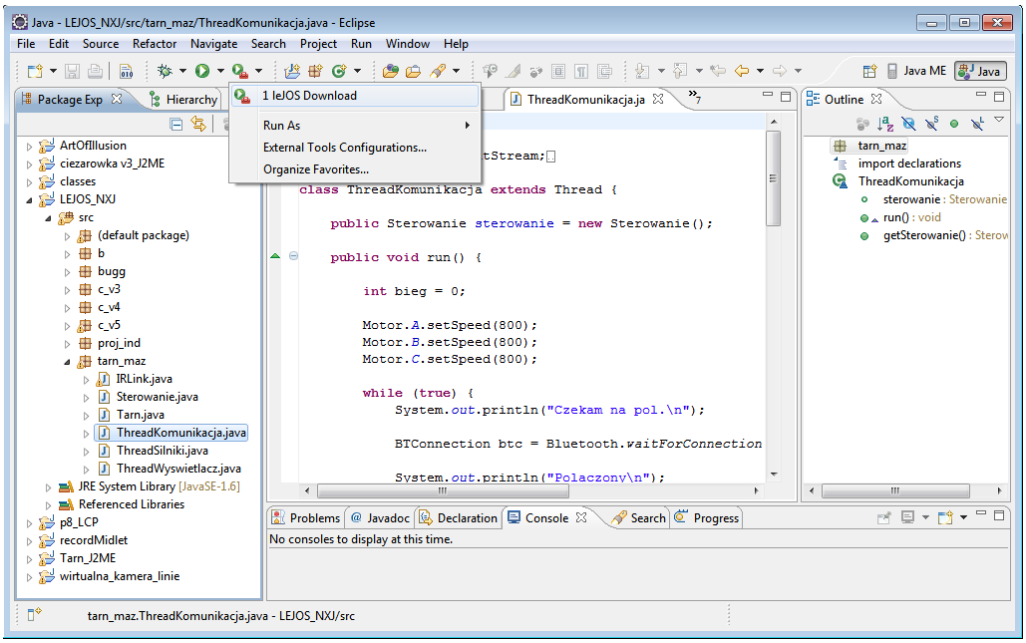
Istnieje środowisko programistyczne Bricx Command Center [10] (rysunek 22, niżej) pozwalające na łatwe pisanie programów NXC. Zostało one stworzone z myślą pisania programów na starszą wersję Mindstorms (RCX), jednak później dodano obsługę nowej kostki NXT.



Rysunek 22. Okno programu Bricx Command Center 3.3.

3.2.4 LeJOS NXJ

LeJOS NXJ [11] to pakiet narzędzi za pomocą których można napisane w Javie programy skompilować przesłać do kostki NXT. Aby móc używać pakietu LeJOS konieczne jest zainstalowanie specjalnego firmware – wirtualną maszynę Java dla kostki NXT. Deweloperzy LeJOS NXJ zapewnili wszystkie potrzebne narzędzia do efektywnego korzystania z tego rozwiązania, w tym bardzo pomocny plugin do Eclipse'a (rysunek 23, niżej) dzięki któremu proces kompilowania i przesyłania programu do kostki jest znacznie ułatwiony. Projekt ten jest stale rozwijany a liczba jego użytkowników rośnie.



Rysunek 23. Okno środowiska programistycznego Eclipse z pluginem LeJOS

3.2.5 Wybór języka

Do wykonania programu na kostkę NXT wybrałem pakiet LeJOS NXJ [11]. Głównymi argumentami przemawiającymi na jego korzyść była znajomość języka Java oraz duża, w porównaniu z innymi rozwiązaniami, ilość materiałów w Internecie. Na stronie LeJOS funkcjonuje aktywne forum, na którym udzielają się autorzy projektu oraz liczni użytkownicy.

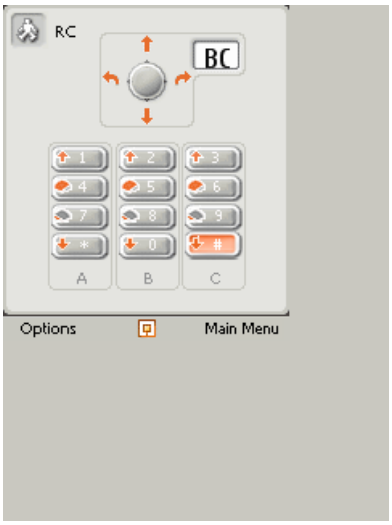
4 Analiza istniejących rozwiązań do zdalnej komunikacji z kostką NXT

Większość powstałych aplikacji do sterowania kostką NXT przeznaczona jest na telefony komórkowe posiadające możliwość uruchamiania aplikacji Java Mobile Edition. Istnieje także możliwość komunikacji pomiędzy kostką a innymi urządzeniami mobilnymi, jak np. palmtopy z systemem Windows Mobile, jednak z uwagi na to, że nie posiadam takiego urządzenia nie zapoznałem się z aplikacjami na tą platformę. Poniżej opiszę trzy aplikacje w technologii Java Mobile Edition, jakie udało mi się przetestować.

4.1 NXTmobile

Jest to aplikacja wydana przez firmę Lego Group [1] niedługo po premierze zestawu Lego Mindstorms NXT w 2006 roku. Komunikuje się z kostką za pomocą protokołu LCP (Lego Communication Protocol), dzięki czemu możliwe jest sterowanie kostkami z firmware Lego, LeJOS, RobotC i innymi obsługującymi ten protokół. Ponadto pozwala uruchamiać programy zapisane na kostce oraz sterować programami NXT-G.

Posiada dużo funkcji jednak większość z nich jest niepotrzebna, a reszta nie do końca przemyślana, przez co korzystanie z tej aplikacji jest niewygodne i nieintuicyjne. Ponadto rozdzielczość interfejsu aplikacji nie jest dostosowana do rozdzielczości 240x320 jaka jest aktualnie najczęściej spotykana w telefonach komórkowych (rysunek 24, niżej).



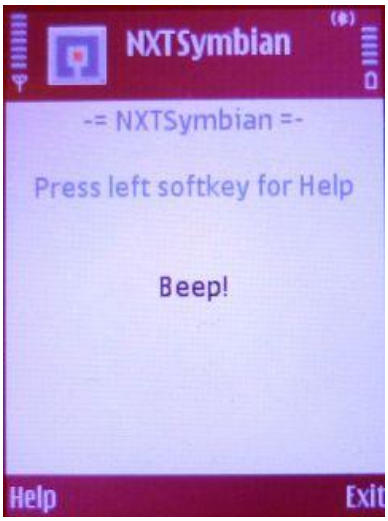
Rysunek 24. Ekran aplikacji NXTMobile.

Aplikacja ta nie jest już prawdopodobnie wspierana, ponieważ nie można jej już pobrać ani uzyskać żadnych informacji na stronie producenta. W dalszym ciągu można ją jednak pobrać na niezależnych stronach, jak np. Blog Lego Robotic [13].

4.2 NXTSymbian

Za pomocą tej aplikacji można sterować zachowaniem programu na kostce NXT napisanym w języku blokowym NXT-G. Program ten, podobnie jak aplikację do sterowania na telefon komórkowy można pobrać ze strony autora - Pedro Miguela [14].

W programie na kostkę możliwe jest przypisanie różnych akcji na 7 komend. Komendy te są wysyłane z urządzenia mobilnego po naciśnięciu odpowiednich klawiszy. Wadą takiego rozwiązania jest ograniczenie do możliwości języka NXT-G oraz brak komend puszczania klawiszy. Ponadto w aplikacji czasem pojawiają się błędy, co skutkuje zamykaniem się programu. Przykładowy ekran tej aplikacji widoczny jest na rysunku 25 (niżej). Zdjęcie to zostało wykonane aparatem fotograficznym, ponieważ NXTSymbian niepoprawnie odświeża ekran, co skutkuje niemożliwością zrobienia zrzutu ekranu.

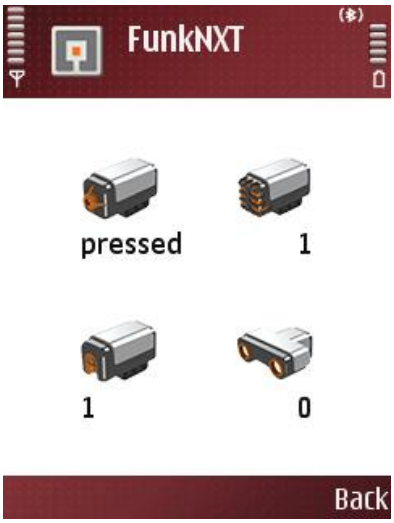


Rysunek 25. Ekran aplikacji NXTSymbian.

4.3 FunkNXT

Jest to dość prosta aplikacja komunikująca się z kostką za pomocą protokołu LCP, dzięki czemu może być używana z firmware Lego jak i innymi obsługującymi ten protokół. Pozwala na niezależne sterowanie trzema silnikami, jednak nie można ich niezależnie zatrzymać. Gdy chcemy zatrzymać jeden, zatrzymują się wszystkie. Oprócz tego występują opóźnienia w komunikacji nawet do ok. 300 ms, co podczas rajdu jest nie do przyjęcia. FunkNXT nie pozwala także wybrać, z którą kostką chcemy się połączyć, po prostu łączy się z pierwszą jaką znajdzie, co może spowodować połączenie się z niepożądaną kostką. Program ten pozwala na obserwację stanu czujników na ekranie telefonu (rysunek 26, niżej). Autorem tej aplikacji jest Mario Patino [15].

Mimo wielu wad, jakie posiada ten program, jest to moim zdaniem najlepszy wybór z gotowych aplikacji do zdalnego sterowania kostką NXT.



Rysunek 26. Ekran sterowania FunkNXT.

4.4 Porównanie

Poniżej znajduje się tabela (tabela 2) z zestawionymi funkcjami i cechami testowanych aplikacji. Ostatnia aplikacja „Tarn" to aplikacja napisana przeze mnie.

	NXTmobile	**NXTSymbian v0.1**	**FunkNXT v0.9.4**	***Tarn v1.2***
Sterowanie robotem typu tribot	tak	tak	tak	*nie*
Niezależnie sterowanie Silnikami	po stworzeniu programu na kostkę	po stworzeniu programu na kostkę	częściowe	*tak*
Wyszukiwanie kostek	tak	tak	tak	*nie*
Wybór danej kostki	tak	tak	nie	*n. d.*
Połączenie z niewidoczną kostką po sparowaniu	nie	nie	tak	*tak*
Wysyłanie komend do programu	tak	tak	nie	*tak*
Długość łączenia (s)	25	75	18	*3*
Wymagany program na kostce	nie	tak	nie	*tak*
Obsługa sensorów	tak	nie	tak	*nie*

Tabela 2. Porównanie aplikacji do zdalnego sterowania kostką NXT.

5 Realizacja projektu

5.1 Opis uruchomieniowy

Aby móc stworzyć niniejsze oprogramowanie niezbędne było zainstalowanie i skonfigurowanie szeregu narzędzi.

5.1.1 Instalacja oprogramowania narzędziowego dla kostki NXT

Poniżej wymienione są w kolejności czynności jakie wykonałem aby móc korzystać ze środowiska Eclipse do pisania i przesyłania programów do kostki NXT.

- Zainstalowałem Java(TM) SE Development Kit 6 pobrany ze strony Sun http://java.sun.com/javase/
- Zainstalowałem środowisko programistyczne Eclipse IDE for Java EE Developers w wersji Galileo pobrane ze strony platformy Eclipse http://www.eclipse.org/
- Zainstalowałem sterownik do kabla USB „LegoMindstormsNXTdriver32.msi” ze strony http://www.robotc.net/download/nxt/
- Zainstalowałem dystrybucję LeJOS NXJ 0.85 ze strony projektu LeJOS [11] w standardowej lokalizacji „C:\Program Files\leJOS NXJ”
- Upewniłem się, że zmienne środowiskowe są prawidłowo ustawione:
  LEJOS_HOME C:\Program Files\leJOS NXJ
  NXJ_HOME C:\Program Files\leJOS NXJ
  PATH C:\Program Files\leJOS NXJ\bin;${NXJ_HOME}/bin:${PATH};
- Zainstalowałem LeJOS NXJ Eclipse Plugin 0.85 korzystając z narzędzia instalacji nowego oprogramowania (Help -> Install New Software, adres URL: http://lejos.sourceforge.net/tools/eclipse/plugin/nxj/)
- Wgrałem firmware LeJOS na kostkę uruchamiając graficzną wersję instalatora nxjflashg.bat znajdującą się w C:\Program Files\leJOS NXJ\bin\
- W programie Eclipse we właściwościach projektu programu na kostkę NXT w opcjach Java Compiler ustawiłem „Java compliance level” na 1.3.
- Skonfigurowałem sposób wysyłania skompilowanego programu do kostki [16]. Wybrałem z paska narzędzi menu Run -> External Tools -> External Tools Configuration. W pole Location wpisałem „C:\Program Files\leJOS NXJ\bin\lejosdl.bat”, w pole Working Directory „${project_loc}\bin”, w Arguments „${java_type_name}”.

Następnie dodałem skrót do uprzednio stworzonej komendy wybierając Run -> External Tools -> Organize Favorites.

Po wykonaniu powyższych kroków możliwe było już wysyłanie aplikacji do kostki zarówno przez kabel USB jak i połączenie Bluetooth (po sparowaniu urządzenia z komputerem) za jednym kliknięciem.

5.1.2 Instalacja oprogramowania narzędziowego do tworzenia aplikacji J2ME na urządzenie mobilne

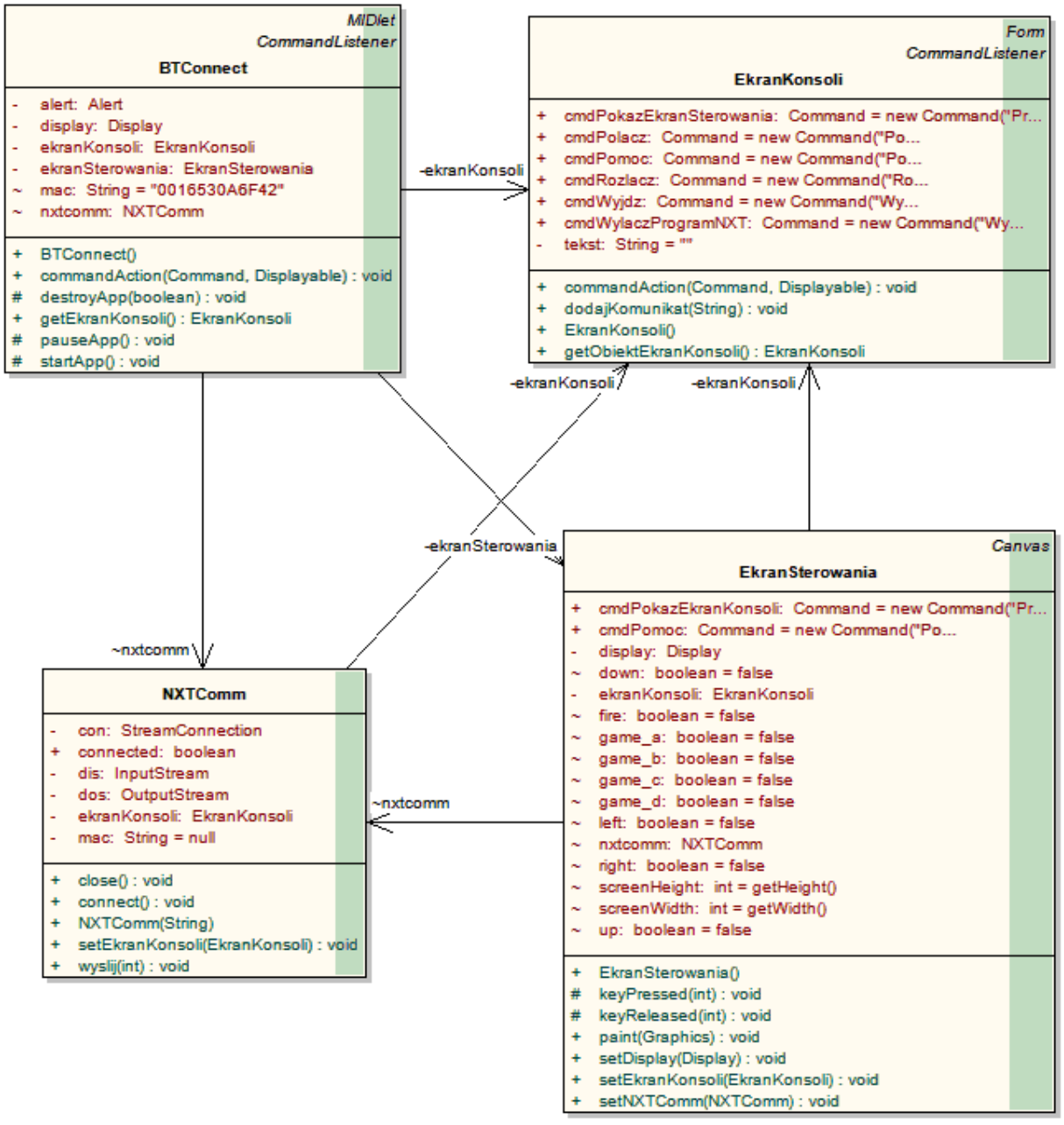
Poniżej wymienione są czynności jakie wykonałem aby móc korzystać ze środowiska Eclipse do pisania i przesyłania programów na telefon komórkowy Nokia z obsługą aplikacji Java.

- Zainstalowałem narzędzie Sun Java Wireless Toolkit for CLDC (wersja 2.5.2_01) ze strony http://java.sun.com/products/sjwtoolkit/
- Zainstalowałem pakiet narzędzi Mobile Tools for Java 1.0.1 korzystając z narzędzia instalacji nowego oprogramowania (Help -> Install New Software, adres URL: http://download.eclipse.org/dsdp/mtj/updates/1.0.1/stable/)
- W programie Eclipse we właściwościach projektu aplikacji na urządzenie mobilne na stronie JavaME dodałem nową konfigurację. Z listy dostępnych zestawów SDK wybrałem Sun Java™ Wireless Toolkit 2.5.2_01 for CLDC

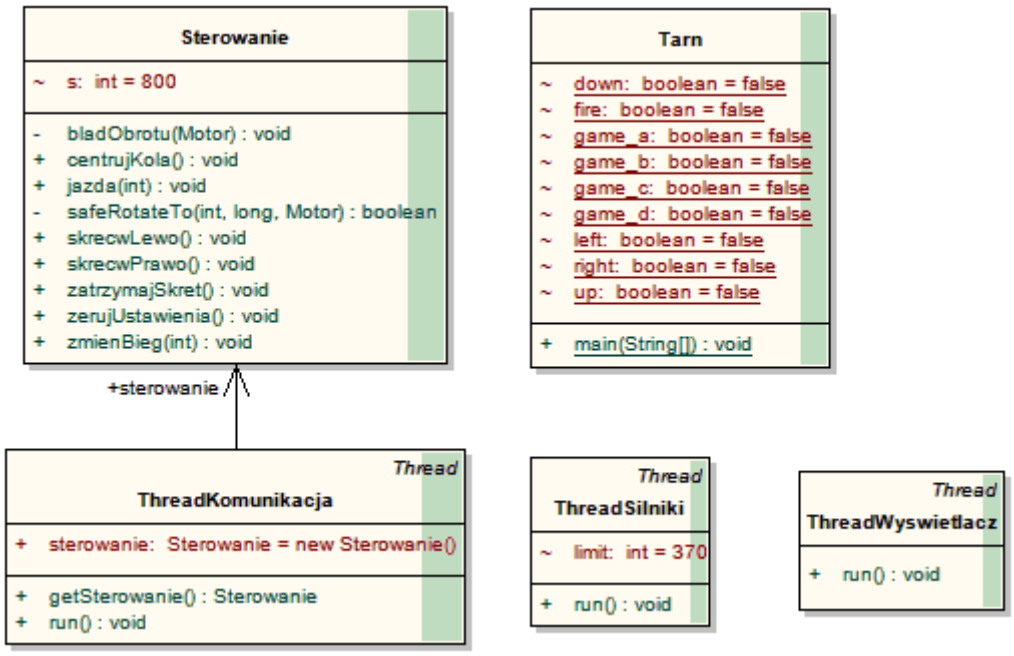
5.2 Opis implementacyjny

5.2.1 Diagramy klas

Poniżej znajdują się diagramy klas aplikacji Tarn przeznaczonej na urządzenie mobilne (rysunek 27) oraz aplikacji na kostkę NXT (rysunek 28, strona 29).



Rysunek 27. Diagram klas aplikacji przeznaczonej na urządzenie mobilne.



Rysunek 28. Diagram klas aplikacji przeznaczonej na kostkę NXT.

5.2.2 Opis metod

W tym rozdziale przedstawione są najciekawsze metody używane w aplikacjach.

- **public void connect() w klasie NXTComm w aplikacji Tarn na telefon komórkowy**

Metoda ta ustanawia połączenie Bluetooth z kostką NXT. Wywołania wszystkich metod związanych z połączeniem są otoczone klauzulą „try/catch” przechwytującą błędy związane z połączeniem.

Adres MAC kostki ustawiany jest w konstruktorze klasy zawierającą tą metodę. Adres URL połączenia jest zapisany na stałe wraz z jego ustawieniami, zmienny może być tylko adres MAC wchodzący w skład adresu połączenia. Metoda ma dostęp do obiektu ekran konsoli dzięki czemu może drukować komunikaty na ekran urządzenie mobilnego.

```
public void connect() {

	if(!connected) {
		try {
			ekranKonsoli.dodajKomunikat("Łączę z " + mac);
			con =(StreamConnection)Connector.open("btspp://" + mac +
				":1;authenticate=false;encrypt=false;master=false");
			dos = con.openOutputStream();
			dis = con.openInputStream();
			ekranKonsoli.dodajKomunikat("Łączenie zakończone");
			connected = true;
		}
		catch (IOException e) {
			ekranKonsoli.dodajKomunikat("Nie można połączyć");
			connected = false;
		}
	}
}
```

Fragment kodu 1. Metoda public void connect().

- **public void run() w klasie ThreadKomunikacja w aplikacji Tarn na NXT**

Metoda odpowiedzialna za działanie wątku threadKomunikacja. Wątek ten odbiera komendy wysyłane przez urządzenie mobilne i uruchamiania odpowiednie metody sterujące pojazdem.

```
public void run() {

	int bieg = 0;
	Motor.A.setSpeed(800);	// silnik skręcający koła
	Motor.C.setSpeed(800);	// silnik napędowy
	while (true) {

		boolean wylaczProgram = false;
		System.out.println("Czekam na pol.\n");
		BTConnection btc = Bluetooth.waitForConnection(0,
			NXTConnection.RAW, null);
		System.out.println("Polaczony\n");
		DataInputStream dis = btc.openDataInputStream();
		DataOutputStream dos = btc.openDataOutputStream();
		try {
			boolean zakonczPolaczenie = false;
			byte n;
			while (!zakonczPolaczenie) {
				n = dis.readByte();
				if (n == 0) sterowanie.centrujKola();
				if (n == 1)  sterowanie.skrecwLewo();
				if (n == 2 || n == 6) sterowanie.zatrzymajSkret();
				if (n == 3)  sterowanie.jazda(-1);
				if (n == 4) sterowanie.jazda(0);
				if (n == 5)  sterowanie.skrecwPrawo();
				if (n == 7)  sterowanie.jazda(1);
				if (n == 9 && bieg < 2)
					sterowanie.zmienBieg(++bieg);	//bieg < 4
				if (n == 10 && bieg > 1) sterowanie.zmienBieg(
					--bieg);
				if (n == 11) {
					sterowanie.zmienBieg(0);
					bieg = 0;
				}
				if (n == 12) zakonczPolaczenie = true;
				if (n == 8) {
					wylaczProgram = true;
					zakonczPolaczenie = true;
				}
			}
		}
		catch (IOException e1) {}

		try {
			dis.close();
			dos.close();

		} catch (IOException e) {}

		try {
			Thread.sleep(100);
			sterowanie.zerujUstawienia();
		}
		catch (InterruptedException e) {}
		btc.close();
		System.out.println("Pol. zakonczone\n");
		if (wylaczProgram)
			System.exit(0);
	}
}
```

Fragment kodu 2. Metoda public void run().

- **private boolean safeRotateTo(int limitAngle, long time, Motor motor) w klasie Sterowanie w aplikacji Tarn na NXT**

Metoda obracająca silnik do ustalonego kąta. Zapobiega blokowaniu się silnika w razie napotkania dużego oporu podczas obracania się poprzez sprecyzowanie maksymalnego czasu jaki może być poświęcony na obrót. W razie upłynięcia ustalonego czasu metoda zwraca wartość „false" a obracający silnik zatrzymuje się (ale nie blokuje). W razie pomyślnego obrotu silnik blokuje się w końcowej pozycji i metoda zwraca wartość „true". Parametry metody to:

- int limitAngle - kąt do jakiego silnik ma się obrócić
- long time - maksymalny czas jaki może być poświęcony na obrót silnika w ms
- Motor motor - silnik jaki ma się obrócić

```
private boolean safeRotateTo(int limitAngle, long time, Motor motor){

	int lockPower = 50;

	motor.setSpeed(s);
	motor.regulateSpeed(true);

	long currTime = System.currentTimeMillis();

	if(limitAngle > motor.getTachoCount()){

		motor.forward();

		while(motor.getTachoCount() < limitAngle){

			if((limitAngle - motor.getTachoCount()) < 70)
				motor.setSpeed(s / 5);
			if(System.currentTimeMillis() > currTime + time){
				bladObrotu(motor);
				return false;
			}
		}
	}

	else {
		motor.backward();

		while(motor.getTachoCount() > limitAngle){

			if((motor.getTachoCount() - limitAngle) < 70)
				motor.setSpeed(s / 5);
			if(System.currentTimeMillis() > currTime + time){
				bladObrotu(motor);
				return false;
			}
		}
	}
	motor.lock(lockPower);
	return true;
}
```

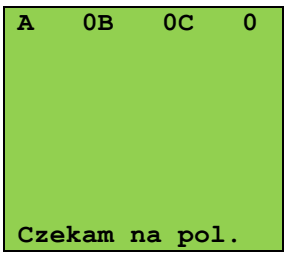
Fragment kodu 3. Metoda private boolean safeRotateTo(int limitAngle, long time, Motor motor).

5.3 Funkcjonalność aplikacji

Przede wszystkim dążyłem do tego, aby aplikacja była szybka i niezawodna. W związku z tym zrezygnowałem z możliwości wyszukiwania różnych kostek na rzecz szybkiego łączenia z się z wcześniej zdefiniowaną po adresie MAC kostką.

5.3.1 Łączenie się z kostką

Aby możliwe było poprawne połączenie z kostką program Tarn.nxj musi być na niej uruchomiony. Program Tarn.nxj po uruchomieniu oczekuje na połączenie (rysunek 29).



Rysunek 29. Program uruchomiony na kostce oczekujący na połączenie.

Widoczne na górze ekranu symbole A, B i C oznaczają wskazania tachometrów poszczególnych silników. Aby wskazania te były aktualne i niezależne od innych wykonywanych czynności wyświetlane są przez oddzielny wątek. Silnik A służy do napędu układu skrętnego, silnik B odpowiada za zmianę przełożeń w skrzyni biegów. Silnik C jest silnikiem napędowym i nie posiada tachometru więc wskazanie tachometru C zawsze jest równe 0 (w pierwotnej wersji silnik C służył jako drugi silnik do obsługi skrzyni biegów). Wskazania tachometrów służą do kontroli, czy silniki zachowują się poprawnie, i czy punkty zerowe silników (pozycja silnika w której wskazanie tachometru wynosi 0) są prawidłowo ustawione.

Przed włączeniem programu na kostce NXT skrętne osie powinny być w centralnym położeniu a w skrzyni biegów włączony bieg jałowy. Dzięki temu po włączeniu programu na kostce punkty zerowe silników będą odpowiadały centralnym pozycjom skrętu i skrzyni biegów.

Gdy program Tarn.nxj jest już uruchomiony na kostce można nawiązać z nią połączenie z telefonu komórkowego. Należy pamiętać, aby na urządzeniu mobilnym Bluetooth był włączony. W innym przypadku wyświetlony zostanie komunikat „Nie można nawiązać połączenia”. Na rysunku 30 (niżej) przedstawiona jest sekwencja czynności potrzebnych do połączenia się z NXT.

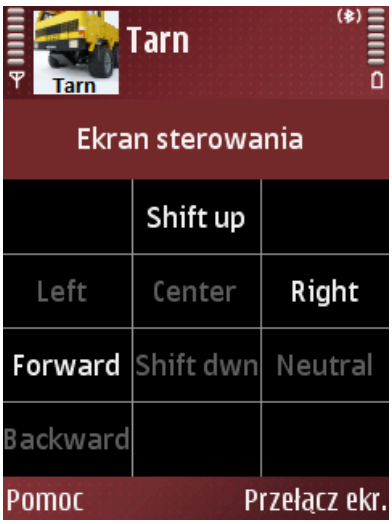
Rysunek 30. Łączenie się z kostką NXT.

Po ekranie nr 4 na rysunku 30 (wyżej) aplikacja automatycznie przełącza się na ekran sterowania, w tym momencie możliwe jest już sterowanie pojazdem.

5.3.2 Sterowanie pojazdem

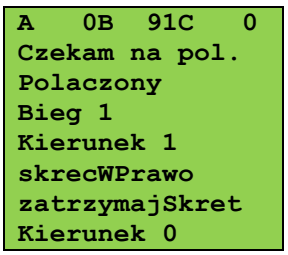
Po ustanowieniu połączenia ekran telefonu wyświetla schemat klawiatury numerycznej wraz z nazwami funkcji, jakie są przypisane poszczególnym klawiszom. Po naciśnięciu jakiegoś przycisku podświetla się on na ekranie na biało, natomiast po puszczeniu jest znów szary. Po naciśnięciu lewego przycisku funkcyjnego „Pomoc” pojawi się okienko pomocy z funkcjami przypisanymi do poszczególnych klawiszy oraz informacja o autorze.

Komendy wysyłane są za pomocą wartości `byte`. Możliwe jest wysyłanie wielu komend równocześnie, nie kolidują one ze sobą w żaden sposób i nie ma konfliktów klawiszy (rysunek 31, niżej). Każde puszczenie przycisku także jest komendą, np. puszczenie przycisku 4/”w lewo” wysyła komendę „zatrzymaj skręt”. Dodatkowo, możliwe jest sterowanie za pomocą klawiszy nawigacyjnych (lewo, prawo, góra, dół i środek), które odpowiadają klawiszom 2, 4, 6, 8 i 5.



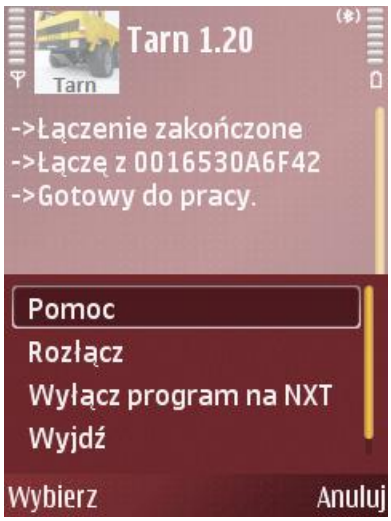
Rysunek 31. Ekran podczas sterowania pojazdem.

Na rysunku 32 (niżej) widać ekran kostki NXT po ustanowieniu połączenia i po otrzymaniu pierwszych komend. Na ekranie wyświetlanych jest 7 ostatnio wykonanych komend oraz aktualne wskazania tachometrów.



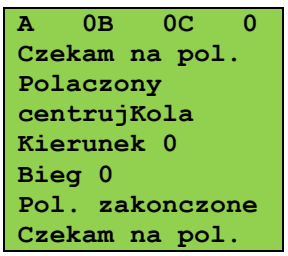
Rysunek 32. Ekran kostki NXT po ustanowieniu połączenia.

Aby powrócić do ekranu konsoli należy nacisnąć prawy przycisk menu „Przełącz ekr." (rysunek 31, wyżej). W ekranie konsoli znajduje się przycisk „Opcje" za pomocą, którego można otworzyć menu z różnymi opcjami wyłączenia programu (rysunek 33, niżej).



Rysunek 33. Ekran konsoli z otwartym menu "Opcje".

Opcja „Rozłącz" zakańcza połączenie z kostką, jednak przed tym wysyła komendę przygotowania pojazdu do rozłączenia. Komenda ta centruje silniki kół oraz skrzyni biegów oraz zatrzymuje silnik napędowy oraz przełącza program w tryb oczekiwania na połączenie (rysunek 34, niżej).



Rysunek 34. Ekran kostki NXT po rozłączeniu.

Opcja „Wyłącz program na NXT” jest podobna do poprzedniej opcji, tyle, że po zakończeniu połączenia wyłącza na kostce program Tarn.nxj. Opcja ta szczególnie przydaje się do testowania programu na kostkę NXT.

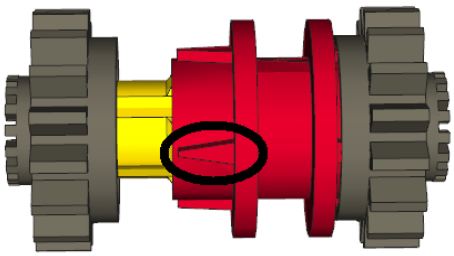
Opcja „Wyjdź” jest podobna do opcji „Rozłącz” tyle, że po rozłączeniu aplikacja na telefonie jest wyłączana. Ta opcja przydatna jest z kolei przy testowaniu aplikacji na urządzenie mobilne.

Zasięg połączenia wynosi ok. 50m w otwartej przestrzeni. Jest to znacznie więcej niż jest wymagane, ponieważ sterowanie pojazdem podczas zawodów odbywa się z co najwyżej kilku metrów.

5.3.3 Zabezpieczenia

Najwrażliwszym elementem pojazdu jest skrzynia biegów, dlatego też musiałem przewidzieć różne scenariusze, jakie mogą wystąpić podczas jej używania. W rozdziale tym pominę mechaniczne zabezpieczenia samej skrzyni, jakie mają na celu zapobieżenie rozpinaniu się jej fizycznie.

Skrzynia działa na zasadzie suwanej przekładni doprowadzającej moment z silnika na różne zębatki o różnych ilościach zębów (rysunek 35, niżej).



Rysunek 35. Element skrzyni biegów. Czerwona suwnica suwa się po żółtej ośce napędzanej przez silnik.

Suwnica zazębia się z zębatkami za pomocą 4 wypustek (wewnątrz szarych zębatek także są 4 takie wypustki). Może się tak zdarzyć, że wypustki trafią akurat na siebie uniemożliwiając zazębienie się suwnicy i zębatki.

Używając standardowej metody void rotateTo(int limitAngle, boolean immediateReturn) do zmiany przełożenia, silnik przesuwający suwnicę blokuje się co uniemożliwia ponowną próbę zmiany biegu.

Jeśli metoda odpowiedzialna za zmianę biegów nie zwraca, jak void rotateTo(int limitAngle), żadnej wartości dopóki bieg nie zostanie zmieniony, mamy wówczas do czynienia z całkowitą utratą kontroli nad pojazdem – auto po prostu wykonuje ostatnie komendy otrzymane przed rozpoczęciem zmiany biegów.

Aby zapobiec tym problemom napisałem metodę boolean safeRotateTo(int limitAngle, Motor motor) (fragment kodu 3, strona 31), która sprawdza, czy po ustalonym czasie bieg został poprawnie zmieniony i jeśli wykryje zablokowanie silnika automatycznie zatrzymuje go umożliwiając dalszą kontrolę nad nim, np. wrzucenie luzu lub ponowną próbę zmiany biegu.

Silnik używany do skręcania kół musi być na tyle silny, aby mógł pokonać duże opory podczas skręcania, szczególnie, że pojazd nie posiada dyferencjałów. Silnik ten mógłby z łatwością zniszczyć układ skrętny, jeśli obróciłby się o kąt większy niż pozwalają na to zwrotnice. W związku z tym zastosowałem zabezpieczenie przed nadmiernym obrotem silnika skrętu. Aby mieć pewność niezależnego od innych czynników działania tego zabezpieczenia realizowane jest ono przez oddzielny wątek.

Ostatnim zabezpieczeniem, opisywanym już wcześniej i o nieco mniejszej wadze, jest automatycznie centrowanie silników po wyłączeniu aplikacji. Przede wszystkim funkcja ta służy wygodzie użytkowania programu (nie trzeba manualnie centrować silników), ale także jest przydatna przy przypadkowym wyłączeniu aplikacji na urządzeniu mobilnym.

6 Podsumowanie i wnioski

Niniejsza praca inżynierska dotyczy zdalnego sterowania pojazdem Lego NXT. Na początku zbudowałem pojazd przystosowany do zdalnego sterowania zgodny z regulaminem zawodów Grand Prix Mazowsza - model ciężarówki Volvo FL12 8x8x4. Ciężarówka ta posiada napęd na wszystkie 8 kół, 4 koła skrętne, pełne niezależne zawieszenie oraz skrzynię biegów.

Dokonałem dokładniej analizy najpopularniejszych rozwiązań do sterowania kostką NXT. Okazało się, że ich funkcjonalność nie jest wystarczająca do sterowania zaawansowanymi funkcjami, jak np. obsługa skrzyni biegów.

Zapoznałem się także z różnymi językami programowania i środowiskami programistycznymi za pomocą których można pisać aplikacje na kostkę NXT. Po wyborze jednego z nich i implementacji okazało się, że był to trafny wybór, ponieważ udało się zrealizować wszystkie wyznaczone cele.

Napisałem dwie aplikacje w języku Java: pierwszą na urządzenie mobilne zgodną ze specyfikacją Java Mobile Edition, drugą na kostkę NXT w technologii LeJOS. Aplikacje te łączą się ze sobą za pośrednictwem połączenia Bluetooth.

Uzyskany model sterowania, mimo, że dość prosty, okazał się być bardzo efektywny i wygodny w użytkowaniu. Aplikacja jest stabilna i odporna na przewidziane zdarzenia zewnętrzne. Przy przesyłaniu komunikatów nie odnotowuje się opóźnień, a zasięg jest w zupełności wystarczający.

Z pewnością użyję tego oprogramowania w nadchodzących zawodach Grand Prix Mazowsza 2010, a także, po zmodyfikowaniu, do sterowania innymi pojazdami.

7 Możliwości rozbudowy

Istnieje kilka kwestii, które można by poprawić lub rozwinąć. Ulepszenia możliwe są przede wszystkim w oprogramowaniu, ponieważ w momencie pisania pracy sądzę, że konstrukcja ciężarówki jest na tyle dopracowana, że trudno było by coś w niej ulepszyć, aby nadal była zgodna z regulaminem zawodów.

Ograniczająca jest możliwość łączenia się z tylko jedną, zapamiętaną wewnątrz aplikacji kostką. Dlatego przydatna byłaby opcja wyszukiwania nowych kostek NXT i możliwość zapamiętania ich adresu MAC. Następnie można by wybrać opcję łączenia się z jedną konkretną kostką wybraną z listy wcześniej zapamiętanych. W takim rozwiązaniu nadal proces łączenia byłby bardzo szybki a aplikacja uniwersalna.

Przydatna byłaby także możliwość podglądu na ekranie urządzenia mobilnego aktualnego stanu pojazdu, jak np. wrzucony bieg czy stan baterii. Aktualnie program na kostce nie wysyła żadnych komunikatów do mobilnego urządzenia sterującego. Nie wiem, czy spowolniłoby to w jakikolwiek sposób komunikację, ale na pewno warto byłoby spróbować to zaimplementować.

Poza powyższymi ulepszeniami do samej aplikacji sterującej możliwe jest wymyślenie wielu rozszerzeń wykorzystujących taki terenowy pojazd, jak np.:

- czujnik ultradźwiękowy do wykrycia czy przejedzie daną przeszkodę
- żyroskop aby jechał prosto nawet po przeszkodach
- automatyczna zmiana biegów na podstawie obrotów silnika napędowego
- automatyczna zmiana biegów przy przechyle pojazdu
- automatyczne parkowanie na podstawie odczytów z czujników

Bibliografia

1. Strona producenta klocków Lego
   http://www.lego.com, ostatni dostęp 20.01.2010
2. Regulamin zawodów Grand Prix Mazowsza
   http://trucktrial.pl/index.php?option=com_content&task=view&id=67&Itemid=33,
   ostatni dostęp 20.01.2010
3. System Lego Power Functions
   http://powerfunctions.lego.com/en-US/Products/powerfunctions/8885.aspx,
   ostatni dostęp 09.01.2010
4. Strona pakietu narzędzi LDraw
   http://www.ldraw.org/, ostatni dostęp 09.01.2010
5. Strona domowa Jennifer Clark
   http://www.jenniferclarkbass.com/lego/hooklift.htm, ostatni dostęp 05.01.2010
6. Strona Philippe E. Hurbain
   http://www.philohome.com/motors/motorcomp.htm, ostatni dostęp 09.01.2010
7. Strona Lego Mindstorms NXT
   http://mindstorms.lego.com/en-us/default.aspx#, ostatni dostęp 12.01.10
8. Strona języka RobotC
   http://www.robotc.net/, ostatni dostęp 12.01.10
9. Strona języka NXC
   http://nxtasy.org/, ostatni dostęp 12.01.10
10. Strona programu Bricx Command Center
    http://bricxcc.sourceforge.net/, ostatni dostęp 21.01.2010
11. Strona projektu LeJOS
    http://lejos.sourceforge.net/, ostatni dostęp 05.01.2010
12. Porównanie rozwiązań do programowania Lego Mindstorms NXT
    http://www.botmag.com/articles/10-31-07_NXT.shtml, ostatni dostęp 23.01.2010
13. Blog Lego Robotic
    http://mikrobot.blogspot.com/2008/10/java-software-for-control-nxt-robot.html,
    ostatni dostęp 23.01.2010
14. Strona domowa aplikacji NXTSymbian
    http://nxt-symbian.sourceforge.net, ostatni dostęp 04.01.2010

15. Strona domowa aplikacji FunkXNT
http://funknxt.homepage.t-online.de/FunkNXT/FunkNXT.html,
ostatni dostęp 04.01.2010
16. Blog Christopha Bartnecka
http://www.bartneck.de/2008/03/04/java-lego-nxt-eclipse-tutorial/,
ostatni dostęp 12.01.10