Sterowanie robotem minisumo

Ł. Małek

Wrocław, 2007

1 Wstęp

Ćwiczenie ma na celu zapoznanie z problematyką sterowania robotami mobilnymi w warunkach ograniczonego, dynamicznie zmieniającego się środowiska przy ograniczonej informacji sensorycznej.

Ćwiczenie realizowane jest w środowisku symulacyjnym Webots w oparciu o uproszczony model robota minisumo.

1.1 Środowisko symulacyjne

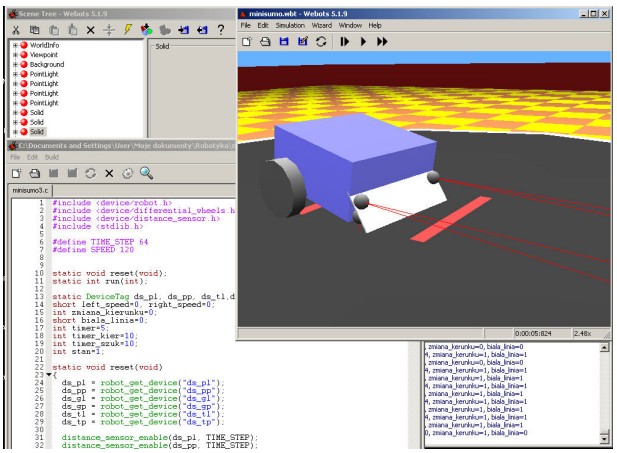
Program Webots umożliwia modelowanie i symulowanie dowolnych robotów. Posiada on rozbudowaną gamę bibliotek gotowych układów sensorycznych. Symulacje generowane w tym systemie uwzględniają właściwości fizyczne modelu dzięki zastosowaniu biblioteki ODE (Open Dynamics Engine). Sterowanie modeli robotów, wykonanych w tym środowisku, może być oprogramowane w językach C, C++ oraz Java.

Po uruchomieniu środowiska komendą

```
webots
```

na ekranie ukażą się cztery okna (Rys. 1):

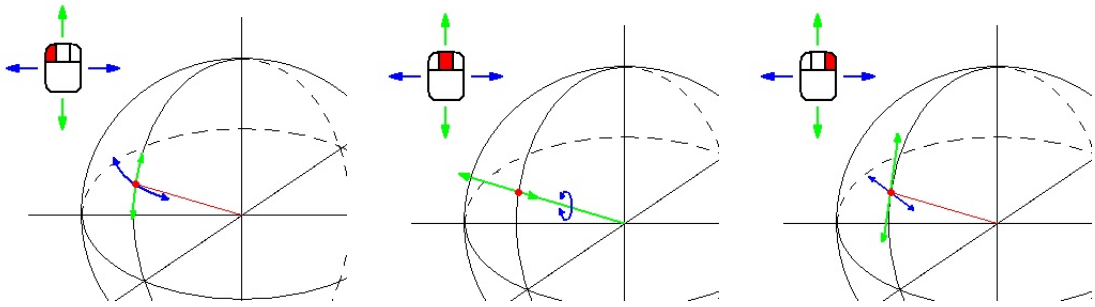
- wizualizacji
- drzewa sceny
- edytor tekstu
- zawartość logów



Rysunek 1: Widok ogólny systemu Webots

Okno wizualizacji jest głównym elementem systemu. Jego zamknięcie powoduje zakończenie działania całego środowiska. Pozwala ono na wczytywanie pliku scen. W przypadku tego ćwiczenia powinien zostać wczytany plik `minisumo.wbt`. Jeśli po uruchomieniu systemu domyślnie otwarta jest inna scena, to należy wczytać właściwą, która znajduje się w katalogu domowym.

Dominującą część okna wizualizacji zajmuje perspektywiczna wizualizacja fragmentu sceny. Służy ono do obserwacji zdarzeń mających miejsce na scenie, oraz do wprowadzania drobnych modyfikacji sceny (zaawansowane i precyzyjne modyfikacje sceny są możliwe jedynie przy wykorzystaniu okna drzewa sceny). Kamera z której widok jest przedstawiony w oknie wizualizacji może zmieniać swoje położenie. Umiejętność sprawnego operowania nią znacząco ułatwia prace w środowisku Webots. Do zmiany położenia kamery oraz jej orientacji względem sceny służy myszka. Do określania położenia i orientacji obiektu w przestrzeni trójwymiarowej służy grupa $SO(3)$, która jest sześciomiarowa. Standardowa mysz umożliwia wykonywanie ruchów jedynie w dwóch ortogonalnych kierunkach po płaszczyźnie stołu więc do wyboru, które współrzędne grupy $SO(3)$ są w danym momencie modyfikowane służy wciśnięty dodatkowo odpowiedni klawisz myszy. Dostępne kombinacje klawiszy oraz ruchów myszy i ich wpływ na zmianę położenia i orientacji kamery przedstawia Rys. 2.



(a) lewy przycisk (b) środkowy przycisk (c) prawy przycisk

Rysunek 2: Sterowanie położeniem i orientacją obserwatora sceny

W sposób analogiczny można sterować położeniem i orientacją elementów znajdujących się na scenie. W tym celu należy zaznaczyć dany obiekt klikając na niego lewym przyciskiem myszy (obiekty zaznaczone stają się półprzeźroczyste, a białą linią uwypuklone stają się ich krawędzie), a następnie trzymając przycisk `Shift` i wykonujemy opisane ruchy które zostały wcześniej przedstawione w przypadku manipulacji kamerą. Metoda ta pozwala na stosunkowo proste, choć niezbyt dokładne manipulacje obiektami. W przypadku gdy chcemy w sposób precyzyjny określić położenie i orientację danego obiektu. Zaznaczamy go, a następnie przechodzimy do okna drzewa sceny, rozwijamy zaznaczony węzeł naciskając na trójkąt znajdujący się po lewej stronie nazwy węzła. Pierwsze dwie górne pozycje: `translation` oraz `rotation` służą odpowiednio do określania położenia oraz orientacji obiektu. Orientacja obiektu określana jest za pomocą obrotu wokół wektora $k = (k_x, k_y, k_x)^T$ o kąt α. Macierz rotacji ma w tym przypadku postać

$$R_{k,\alpha} = \begin{bmatrix} k_x^2(1-c_\alpha)+c_\alpha & k_xk_y(1-c_\alpha)-k_zs_\alpha & k_xk_z(1-c_\alpha)+k_ys_\alpha \\ k_xk_y(1-c_\alpha)+k_zs_\alpha & k_y^2(1-c_\alpha)+c_\alpha & k_yk_z(1-c_\alpha)-k_xc_\alpha \\ k_xk_z(1-c_\alpha)-k_ys_\alpha & k_yk_z(1-c_\alpha)+k_xs_\alpha & k_z^2(1-c_\alpha)+c_\alpha \end{bmatrix},$$

gdzie $c_\alpha = \cos(\alpha)$, $s_\alpha = \sin(\alpha)$.

Najistotniejszą właściwością okna wizualizacji jest umożliwienie uruchomienia symulacji a następnie jej zatrzymanie (symulacje nie zatrzymują się automatycznie!). Funkcje z tym związane znajdują się w menu `Simulation` jak i również są powtórzone w postaci intuicyjnych przycisków, po najechaniu na które wyświetlana zostaje podpowiedz dotycząca ich działania.

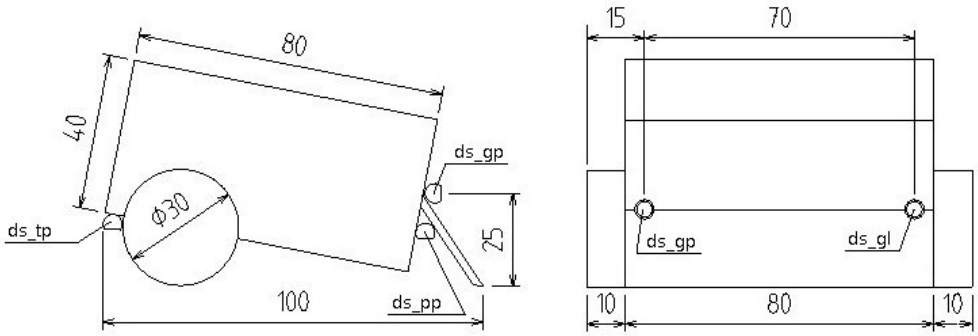
Kolejną istotną zakładką jest wchodzącą w skład paska menu okna symulacji jest `Wizard`. Znajdują się tam pozycje umożliwiające tworzenie nowego projektu oraz nowego sterownika, ta druga będzie wykorzystywana kilkakrotnie podczas ćwiczeń.

Zakładka `Window` przydaje się w momencie gdy zamkniemy któreś z pozostałych okienek (edytor tekstu, log, drzewo sceny), a następnie będziemy potrzebowali je odtworzyć. W tym celu wybieramy z tego menu odpowiednią pozycję.

Dodatkowych informacji na temat pracy ze środowiskiem uzyskamy w menu `Help`.

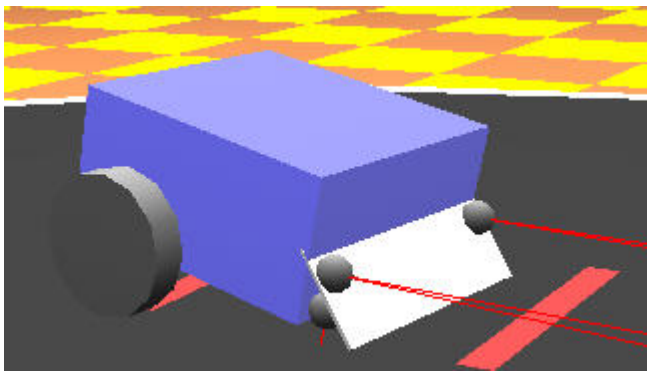
1.2 Robot

W tym ćwiczeniu obiektem naszych zainteresowań jest model prostego robota klasy (2,0) Jego schemat przedstawia Rys 3 Wymiary robota zostały dobrane tak, aby spełniał on



Rysunek 3: Schemat robota minisumo

przepisy regulaminu zawodów minisumo, tzn. aby mieścił się on w prostopadłościanie o podstawie kwadraty o wymiarach 10cm × 10cm. Waga całkowita robota wynosi 0.5kg. Z czego 0.05kg to waga każdego z kół, a pozostałe 0.4kg przypada na korpus. Masa rozłożona jest jednorodnie na całej objętości brył tworzących jego konstrukcję.

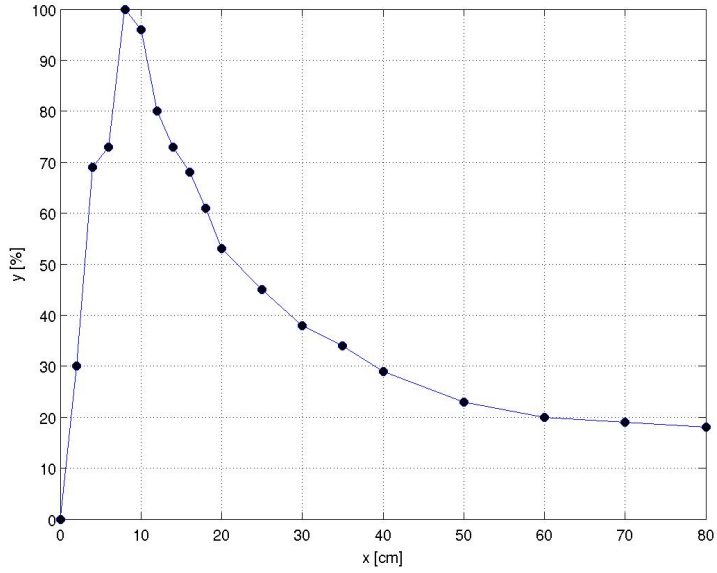


Rysunek 4: Model robota minisumo.

Napęd robota stanowią dwa niezależnie napędzane koła, których oś jest przesunięta do tyłu względem osi symetrii głównej bryły robota. Dzięki temu robot nie ma tendencji do kołysania się podczas zmian prędkości. Trzeci punkt podparcia stanowi znajdujący się z przodu robota pług.

Układ sensoryczny rozważanej platformy mobilnej jest bardzo ograniczony. W jego skład wchodzą 4 czujniki (ds_tp, ds_tl, ds_pp, ds_pl) podczerwieni skierowane w dół służące do

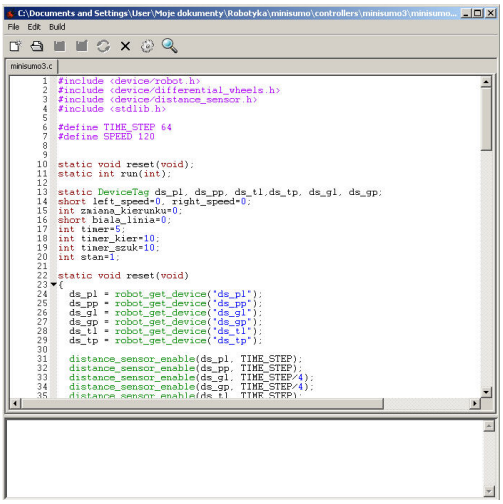
wykrywania białych linia, są one umieszczone w czterech rogach robota. Ostatnie litery w ich symbolach oznaczają poleżenie czujnika. Pierwsza z tych liter oznacza **p**rzód albo **t**ył. Z kolei ostatnia pochodzi od **p**rawy alb **l**ewy. Robot posiada także dwa czujniki odległości skierowane poziomo na wprost robota (ds_gp, ds_gl). Czujniki te posiadają nieliniową charakterystykę która przedstawiona jest na Rys. 5.



Rysunek 5: Zależność wskazań czujników odległości od dystansu do mierzonego obiektu

1.3 Programowanie robota

Zachowanie wirtualnego robota na scenie jest określone przez program kontrolera przypisanego do danego modelu. Każdy model robota znajdujący się na scenie może mieć przypisany swój własny indywidualny kontroler. Kontroler jest programem napisanym w jednym z dopuszczalnych przez system języków w sposób zgodny ze specyfikacją systemu. Niniejsze ćwiczenie powinno być realizowane w oparciu o język C z wykorzystaniem domyślnego edytora teksu wchodzącego w skład systemu Webots (Rys. 6). Edytor ten pomimo swojej prostoty posiada wszystkie najistotniejsze funkcje (kolorowanie składni, numerowanie linii, skok do linii, okno wyniku kompilacji, przyciski kompilacji). Chociaż istnieje spora dowolność w kodzie źródłowym kontrolera dla systemu Webots **zaleca** się wykorzystanie wzorca znajdującego się w katalogu `minisumo/controlers/minisumo0/minisumo0.c`, plik ten wygląda następująco:



Rysunek 6: Okno edytora tekstu

```
#include <device/robot.h>
#include <device/differential_wheels.h>
#include <device/distance_sensor.h>
/*
 * W tym miejscu można dodać inne potrzebne biblioteki.
 */

#define TIME_STEP 64   /* krok cyklu w milisekundach */
#define SPEED 120      /* maksymalna predkość kół    */
/*
 * Tu należy wprowadzać inne potrzebne makrodefinicje.
 */

static void reset(void);
static int run(int);
/*
 * Tutaj należy umieścić prototypy funcji, które będzie się używać
 * w programie.
 */

static DeviceTag ds_pl, ds_pp, ds_tl, ds_tp, ds_gl, ds_gp;
/*
 * Tutaj należy wprowadzić definicję zmiennych globalnych DeviceTag,
 * które są wskaznikami do odpowiednich użądzeń robota. Należy używać
 * słowa kluczowego ''static'' w odniesieniu do tych zmiennych.
 */

/*
 * Funkcja reset jest wywoływana tylko raz podczas inicjalizacji
 * robota.
 */
static void reset(void)
{
  /*
   * Tutaj umeszczamy funkcje ''robot_get_device()'' aby uzyskac
   * wskazniki do odpowiednich uzaczan robota (DeviceTag), które będą
   * wykorzystywane w głównej pętli controli.
   */
  ds_pl = robot_get_device("ds_pl");
  ds_pp = robot_get_device("ds_pp");
  ds_gl = robot_get_device("ds_gl");
  ds_gp = robot_get_device("ds_gp");
  ds_tl = robot_get_device("ds_tl");
  ds_tp = robot_get_device("ds_tp");

  /*
   * Tutaj określamy częśtotliwość próbkowania czujników
   */
  distance_sensor_enable(ds_pl, TIME_STEP);
  distance_sensor_enable(ds_pp, TIME_STEP);
  distance_sensor_enable(ds_gl, TIME_STEP);
  distance_sensor_enable(ds_gp, TIME_STEP);
  distance_sensor_enable(ds_tl, TIME_STEP);
  distance_sensor_enable(ds_tp, TIME_STEP);

  return;
}
```

```
/*
 * To jest funkcja główna pętla kontroli, jest ona wywoływana
 * cyklicznie przez system Webots
 */
static int run(int ms)
{
  unsigned short ds_pl_value;
  unsigned short ds_pp_value;
  unsigned short ds_gl_value;
  unsigned short ds_gp_value;
  unsigned short ds_tl_value;
  unsigned short ds_tp_value;

  /*
   * Wprowadz tutaj funkcje od odczytu danych z sensorów za pomocą:
   * unsigned short distance_sensor_get_value  (DeviceTag sensor);
   */
  ds_pl_value = distance_sensor_get_value(ds_pl);
  ds_pp_value = distance_sensor_get_value(ds_pp);
  ds_gl_value = distance_sensor_get_value(ds_gl);
  ds_gp_value = distance_sensor_get_value(ds_gp);
  ds_tl_value = distance_sensor_get_value(ds_tl);
  ds_tp_value = distance_sensor_get_value(ds_tp);

  /*
   * Tutaj przetwórz dane z sensorów.
   */

  /*
   * Tutaj wprowadz funkcje wysyłającą informację do elementów
   * wykonawczych np.:
   * void differential_wheels_set_speed(short left, short right);
   */
  differential_wheels_set_speed( 0, 0);

  return TIME_STEP; /* to jest wartość kroku w milisekundach */
}

/*
 * To jest główna część programu, która ustawia funkcję inicjalizacji
 * i główną pętlę programu
 */
int main()
{
  robot_live(reset); /* inicjalizacja */
  robot_run(run);    /* start kontrolera */

  return 0;
}
```

Wskazówka: Przydatną funkcją podczas testowania działania algorytmów jest

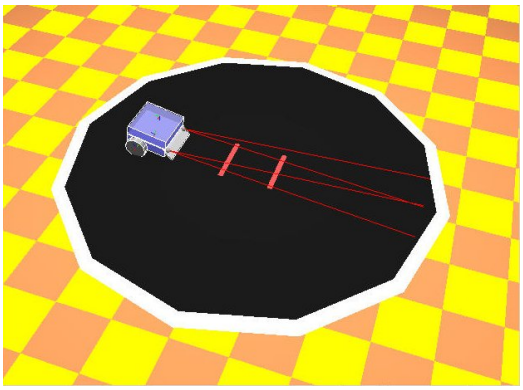
`void robot_console_printf(const char *format, ...);`

która pozwala na wyświetlanie, w sposób analogiczny jak to ma miejsce w przypadku standardowej funkcji `printf`, z tą różnicą, że łańcuch tekstu jest wyświetlany w oknie logów.

2 Zadanie 1

Przygotowanie zadania:

1. Z paska menu okna wizualizacji wybrać `Wizard` → `New robot controller...`
2. Wybrać język C
3. Nazwać kontroler `minisumo1`
4. W oknie drzewa sceny klikając na trójkąt rozwinąć węzeł `DifferentialWheels`
5. Zaznaczyć pozycję `controller` i w prawej części okna kliknąć na przycisk `...`.
6. Wybrać kontroler `minisumo1`[1], a następnie za pomocą Edit... otworzyć edytor tekstów.



Rysunek 7: Robot poruszający się po średnicy ringu

Zadania do wykonania:

1. Zaprogramować[2] robota w taki sposób aby odczytywane wartości czujników linii (ds_tl, ds_tp, ds_pl, ds_pp) były wyświetlane na okno logów.
2. Odczytać wskazania czujników linii w momencie gdy skierowane (kierunek w który są zwrócone czujniki określają czerwone linie) są one na podłoże koloru czarnego.
3. Za pomocą myszki przestawić robota, tak aby jego czujniki znajdowały się nad białą linia otaczającą ring, a następnie odczytać wskazania czujników.
4. Określić poziom wskazań pozwalający na wykrywanie białej linii.
5. Ustawić robota w centrum ringu, a następnie zaprogramować go tak, aby poruszał się on po linii prostej aż do momentu wykrycia białej linii, po czym powinien on zmienić kierunek jazdy na przeciwny. Działanie algorytmu zaprezentować prowadzącemu zajęcia.

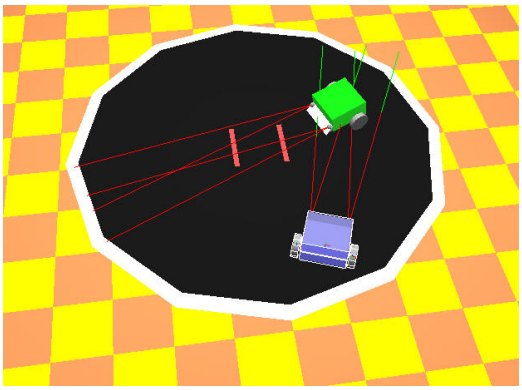
[1] Po każdorazowej zmianie kontrolera należy zapisać scenę naciskając dyskietkę w oknie wizualizacji, po czym zrestartować scenę naciskając naciskając przycisk zawierający dwie strzałki tworzące koło.

[2] Do kompilacji programu służy ikonka koła zębatego znajdująca się w oknie edytora tekstu.

3 Zadanie 2

Przygotowanie zadania:

1. Zwinąć otwarty węzeł `DifferentialWheels`
2. Dodać do sceny drugiego robota naciskając w oknie drzewa sceny ikonę Import (dyskietka ze strzałką skierowaną w lewą stronę).
3. Z katalogu `minisumo/worlds` wybrać plik `sumita.wbt` i nacisnąć przycisk `Open`
4. Rozwinąć nowo dodany (ostatni) węzeł `DifferentialWheels` i wybrać ustawić jego kontroler jako `void`.
5. Umieścić nowo dodanego robota (kolor zielony) w centrum ringu.
6. Umieścić starego robota (kolor niebieski) na skraju ringu tak aby roboty znajdowały się w pozycji jak na Rys. 8
7. Dodać nowy kontroler o nazwie `minisumo2`.
8. Ustawić aby robot niebieski sterowany był przez kontroler `minisumo2`.



Rysunek 8: Robot niebieski śledzi ruch zielonego robota poruszającego się po średnicy ringu.

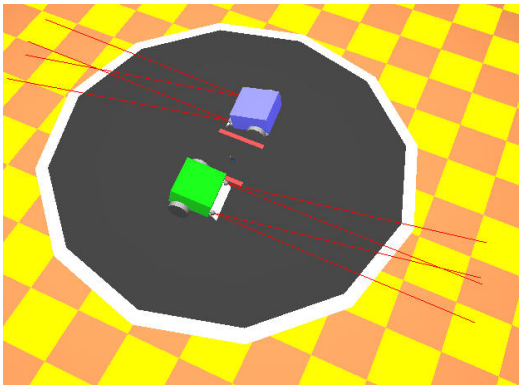
Zadania do wykonania:

1. Zaprogramować robota niebieskiego aby pobierał on wskazania czujników odległości (ds_gl, ds_gp), i wyświetlał je w terminalu logów.
2. Odczytać wskazania czujników odległości przy różnych odległościach do przeszkody. Jako przeszkodę wykorzystać zielonego robota. Potwierdzić nieliniowy charakter czujników linii.
3. Ustawić kontroler robota zielonego na `minisumo1` a następnie umieścić robota jak na Rys. 8.
4. Zaprogramować robota niebieskiego tak, aby śledził on ruch robota zielonego poruszającego się po średnicy ringu. Robot niebieski ustawiać się zawsze przodem do robota zielonego. Działanie algorytmu zaprezentować prowadzącemu zajęcia.

4 Zadanie 3

Przygotowanie zadania:

1. Utworzyć nowy kontroler `minisumo3` i wybrać go dla robota niebieskiego



Rysunek 9: Robot niebieski wypycha robota zielonego z ringu.

Zadanie do wykonania:

1. Zaprogramować robota niebieskiego tak aby zlokalizował robota zielonego i zepchnął go z ringu a sam z niego nie zjechał. Działanie algorytmu zaprezentować prowadzącemu zajęcia.

2. Zmienić kontroler robota zielonego na kontroler `minisumo3`, ustawić roboty tak jak na Rys. 9 a następnie uruchomić symulacje.