

Sprawozdanie Lab 11

Andrzej Żaba, gr_lab 4, nr indeksu: 401490

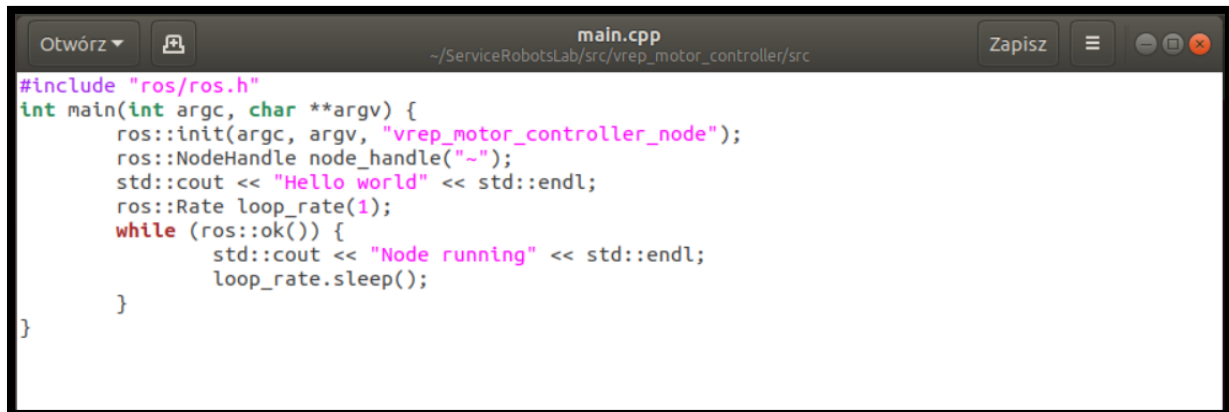
Korzystając z poniższych komend stworzono przestrzeń roboczą w folderze *ServiceRobotsLab* i wstępnie ją skompilowano.

```
andrzej@andrzej-VirtualBox: ~/ServiceRobotsLab
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
andrzej@andrzej-VirtualBox:~$ mkdir ~/ServiceRobotsLab
andrzej@andrzej-VirtualBox:~$ mkdir ~/ServiceRobotsLab/src
andrzej@andrzej-VirtualBox:~$ cd ~/ServiceRobotsLab/src
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ catkin_init_workspace
Creating symlink "/home/andrzej/ServiceRobotsLab/src/CMakeLists.txt" pointing to
"/opt/ros/melodic/share/catkin/cmake/toplevel.cmake"
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ catkin_make
Base path: /home/andrzej/ServiceRobotsLab/src
The specified source space "/home/andrzej/ServiceRobotsLab/src/src" does not exist
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ cd ~/ServiceRobotsLab/src
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ catkin_make
Base path: /home/andrzej/ServiceRobotsLab/src
The specified source space "/home/andrzej/ServiceRobotsLab/src/src" does not exist
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ cd ~/ServiceRobotsLab
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ catkin_make
Base path: /home/andrzej/ServiceRobotsLab
Source space: /home/andrzej/ServiceRobotsLab/src
Build space: /home/andrzej/ServiceRobotsLab/build
Devel space: /home/andrzej/ServiceRobotsLab/devel
Install space: /home/andrzej/ServiceRobotsLab/install
####
#### Running command: "cmake /home/andrzej/ServiceRobotsLab/src -DCATKIN_DEVEL_PREFIX=/home/andrzej/ServiceRobotsLab/devel -DCMAKE_INSTALL_PREFIX=/home/andrzej/ServiceRobotsLab/install -G Unix Makefiles" in "/home/andrzej/ServiceRobotsLab/build"
####
-- The C compiler identification is GNU 7.5.0
-- The CXX compiler identification is GNU 7.5.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
```

Utworzono plik *main.cpp* w którym dalej pisano kod.

```
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ catkin_create_pkg vrep_motor_controller roscpp
Created file vrep_motor_controller/CMakeLists.txt
Created file vrep_motor_controller/package.xml
Created folder vrep_motor_controller/include/vrep_motor_controller
Created folder vrep_motor_controller/src
Successfully created files in /home/andrzej/ServiceRobotsLab/src/vrep_motor_controller. Please adjust the values in anand
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ touch ~/ServiceRobotsLab/src/vrep_motor_controller/src/main.cpp
touch: nie można dotknąć '/home/andrzej/ServiceRobotsLab/src/vrep_motor_controller/src/main.cpp': Nie ma takiego pliku ani katalogu
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$ touch ~/ServiceRobotsLab/src/vrep_motor_controller/src/main.cpp
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab/src$
```

Napisano przykładowy kod w celu sprawdzenia, czy poprzednie kroki zostały wykonane poprawnie.



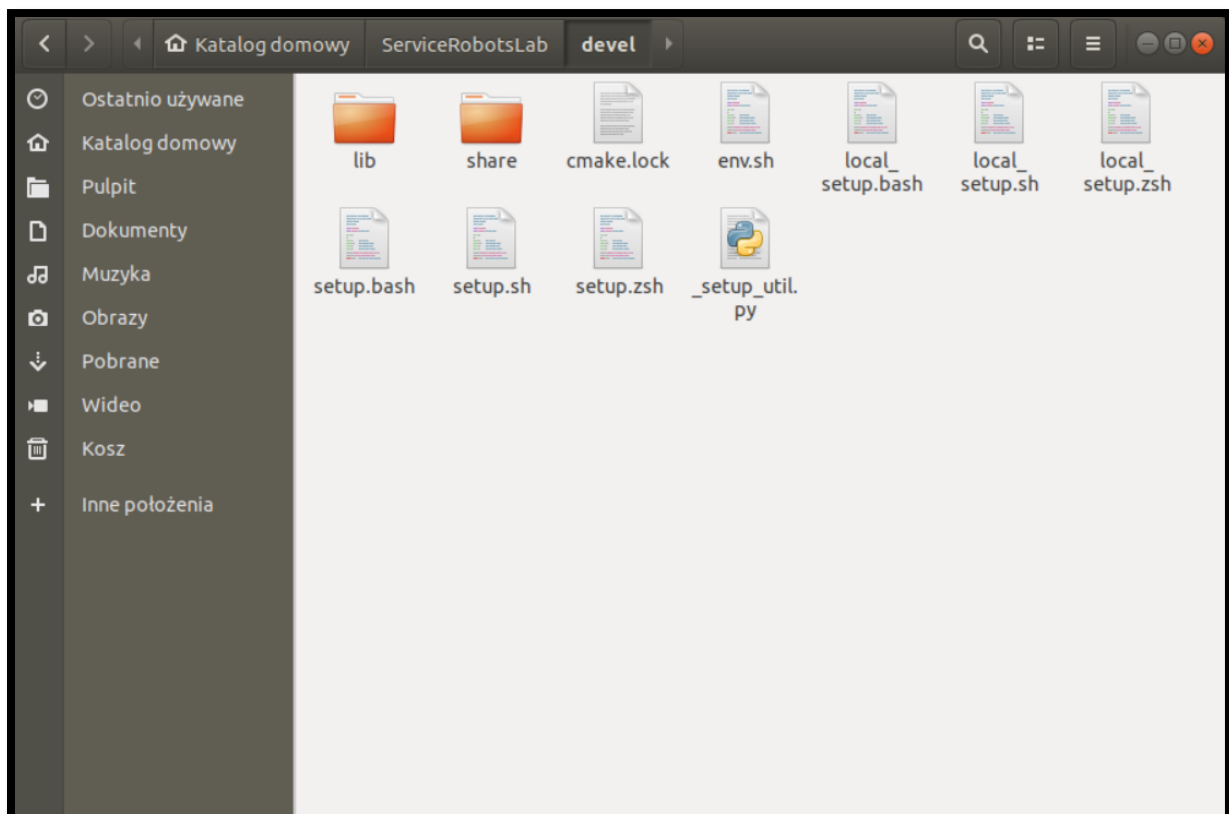
```
#include "ros/ros.h"
int main(int argc, char **argv) {
    ros::init(argc, argv, "vrep_motor_controller_node");
    ros::NodeHandle node_handle("~");
    std::cout << "Hello world" << std::endl;
    ros::Rate loop_rate(1);
    while (ros::ok()) {
        std::cout << "Node running" << std::endl;
        loop_rate.sleep();
    }
}
```

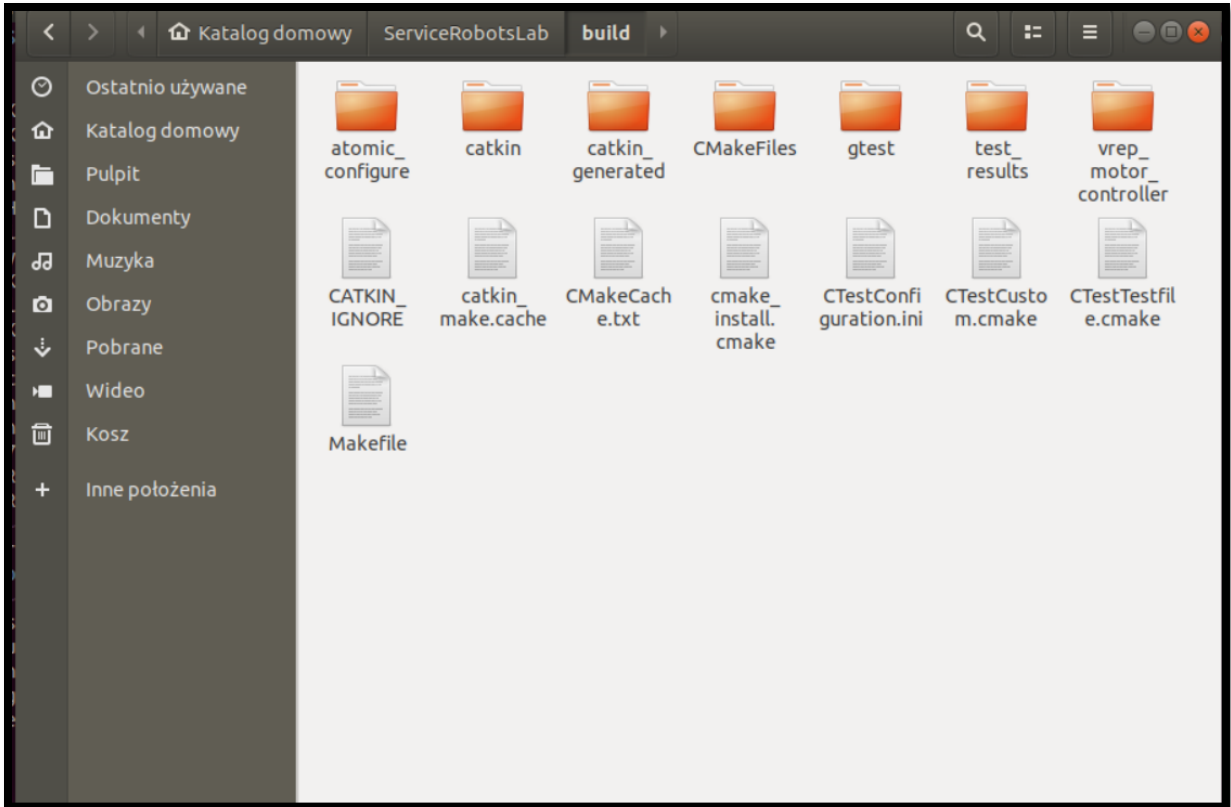
Poddano przestrzeń roboczą kompilacji.

```
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ catkin_make
Base path: /home/andrzej/ServiceRobotsLab
Source space: /home/andrzej/ServiceRobotsLab/src
Build space: /home/andrzej/ServiceRobotsLab/build
Devel space: /home/andrzej/ServiceRobotsLab/devel
Install space: /home/andrzej/ServiceRobotsLab/install
####
#### Running command: "cmake /home/andrzej/ServiceRobotsLab/src -DCATKIN_DEVEL_PREFIX=/home/andrzej/ServiceRobotsLab/devel -DCMAKE_INSTALL_PREFIX=/home/andrzej/ServiceRobotsLab/install -G Unix Makefiles" in "/home/andrzej/ServiceRobotsLab/build"
####
-- Using CATKIN_DEVEL_PREFIX: /home/andrzej/ServiceRobotsLab/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/melodic
-- This workspace overlays: /opt/ros/melodic
-- Found PythonInterp: /usr/bin/python2 (found suitable version "2.7.17", minimum required is "2")
-- Using PYTHON_EXECUTABLE: /usr/bin/python2
-- Using Debian Python package layout
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/andrzej/ServiceRobotsLab/build/test_results
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Found gmock sources under '/usr/src/gtest': gmock will be built
-- Found PythonInterp: /usr/bin/python2 (found version "2.7.17")
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.29
-- BUILD_SHARED_LIBS is on
-- BUILD_SHARED_LIBS is on
-- ~~~ traversing 1 packages in topological order:
-- ~~~ - vrep_motor_controller
-- ~~~
+++ processing catkin package: 'vrep_motor_controller'
==> add_subdirectory(vrep_motor_controller)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/andrzej/ServiceRobotsLab/build
```

```
-- traversing 1 packages in topological order:
--   - vrep_motor_controller
--
-- +++ processing catkin package: 'vrep_motor_controller'
-- ==> add_subdirectory(vrep_motor_controller)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/andrzej/ServiceRobotsLab/build
####
#### Running command: "make -j1 -l1" in "/home/andrzej/ServiceRobotsLab/build"
####
Scanning dependencies of target vrep_motor_controller_node
[ 50%] Building CXX object vrep_motor_controller/CMakeFiles/vrep_motor_controller_node.dir/src/main.cpp.o
[100%] Linking CXX executable /home/andrzej/ServiceRobotsLab/devel/lib/vrep_motor_controller/vrep_motor_controller_node
[100%] Built target vrep_motor_controller_node
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$
```

Stwierdzono poprawność wykorzystanych komend.





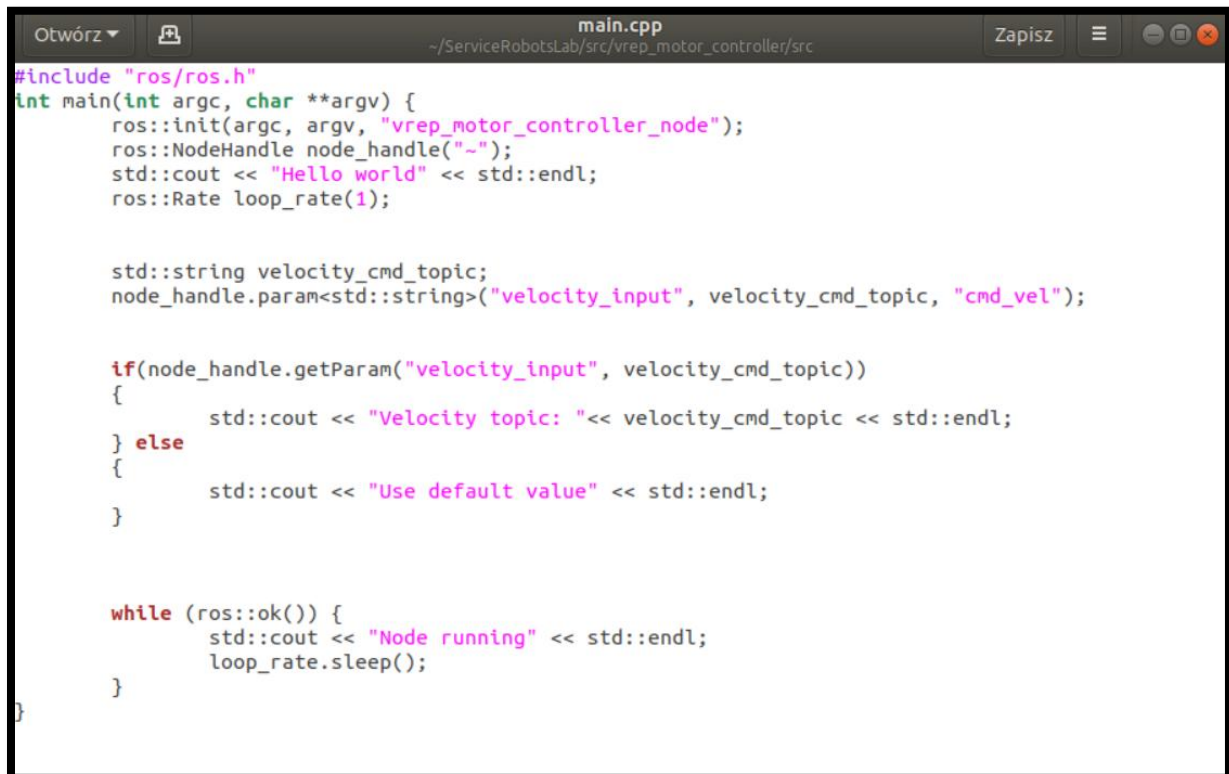
Sprawdzono, czy wszystkie niezbędne pliki zostały utworzone i wprowadzono odpowiednie modyfikacje kodu w pliku *CMakeLists.txt*.

Raz jeszcze skompilowano i uruchomiono wcześniej napisany program.

[illegible]

Stwierdzono poprawne działanie do tej pory napisanego programu.

Dodano nowy kod do pliku main.cpp



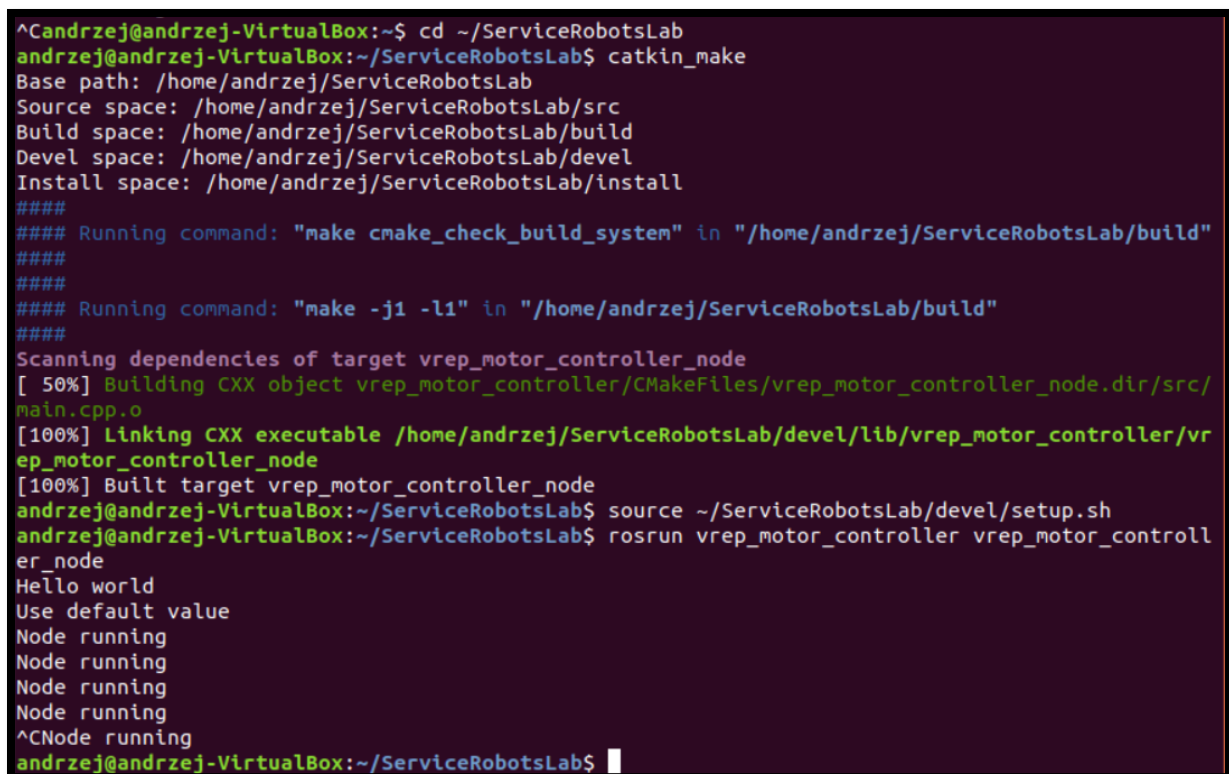
```
#include "ros/ros.h"
int main(int argc, char **argv) {
    ros::init(argc, argv, "vrep_motor_controller_node");
    ros::NodeHandle node_handle("~");
    std::cout << "Hello world" << std::endl;
    ros::Rate loop_rate(1);

    std::string velocity_cmd_topic;
    node_handle.param<std::string>("velocity_input", velocity_cmd_topic, "cmd_vel");

    if(node_handle.getParam("velocity_input", velocity_cmd_topic))
    {
        std::cout << "Velocity topic: " << velocity_cmd_topic << std::endl;
    } else
    {
        std::cout << "Use default value" << std::endl;
    }

    while (ros::ok()) {
        std::cout << "Node running" << std::endl;
        loop_rate.sleep();
    }
}
```

Dalej, skompilowano i uruchomiono program.



```
^Candrzej@andrzej-VirtualBox:~$ cd ~/ServiceRobotsLab
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ catkin_make
Base path: /home/andrzej/ServiceRobotsLab
Source space: /home/andrzej/ServiceRobotsLab/src
Build space: /home/andrzej/ServiceRobotsLab/build
Devel space: /home/andrzej/ServiceRobotsLab/devel
Install space: /home/andrzej/ServiceRobotsLab/install
####
#### Running command: "make cmake_check_build_system" in "/home/andrzej/ServiceRobotsLab/build"
####
####
#### Running command: "make -j1 -l1" in "/home/andrzej/ServiceRobotsLab/build"
####
Scanning dependencies of target vrep_motor_controller_node
[ 50%] Building CXX object vrep_motor_controller/CMakeFiles/vrep_motor_controller_node.dir/src/main.cpp.o
[100%] Linking CXX executable /home/andrzej/ServiceRobotsLab/devel/lib/vrep_motor_controller/vrep_motor_controller_node
[100%] Built target vrep_motor_controller_node
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ source ~/ServiceRobotsLab/devel/setup.sh
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ rosrn vrep_motor_controller vrep_motor_controll
er_node
Hello world
Use default value
Node running
Node running
Node running
Node running
^CNode running
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$
```

Następnie do komendy wywołującej uruchomienie węzła dodano wartość parametru *velocity_input*.

```
^Candrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ rosrn vrep_motor_controller vrep_motor_controller_node
velocity_input:=nazwa_tematu
Hello world x
Velocity topic: nazwa_tematu
Node running
Node running
Node running
Node running
Node running
^CNode running
```

Stwierdzono, że program działa poprawnie.

Zad 1.

Uruchomiono węzły komendami:

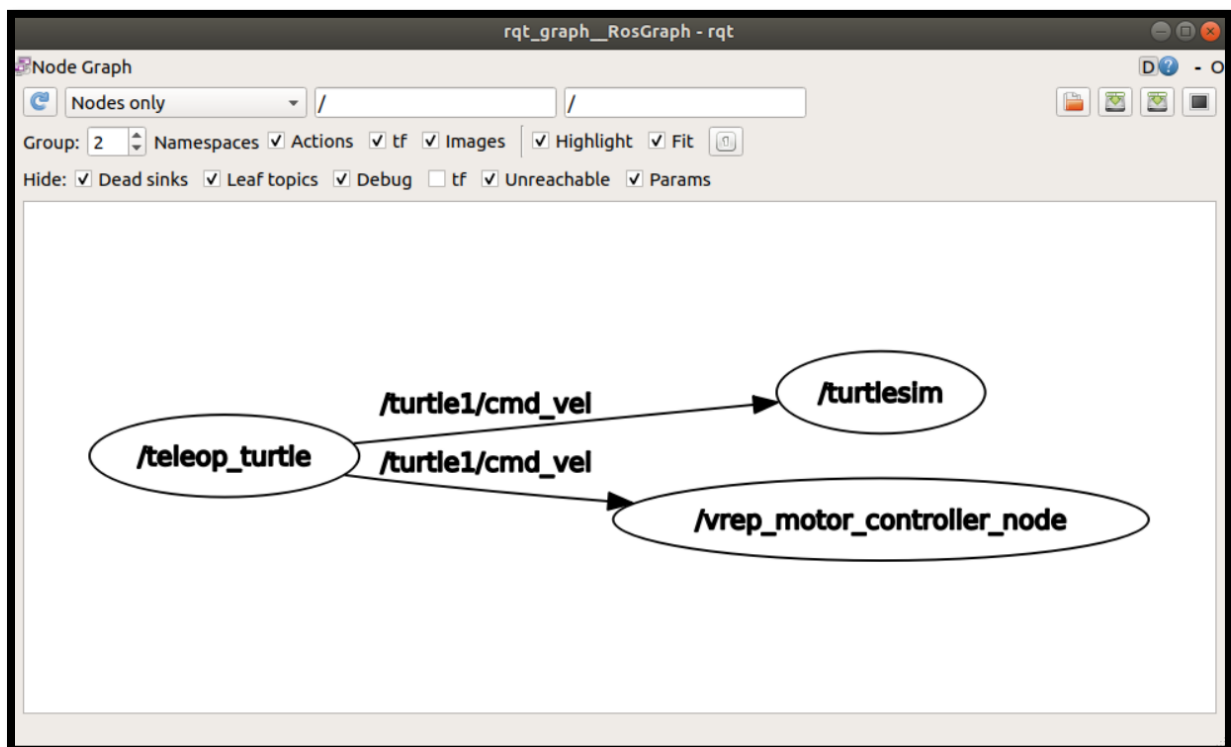
```
$ rosrn turtlesim turtlesim_node
```

```
$ rosrn turtlesim turtle_teleop_key
```

```
$ rosrn vrep_motor_controller vrep_motor_controller_node _velocity_input:=/turtle1/cmd_vel
```

Sprawdzono graf rqt komendą:

```
$ rosrn rqt_graph rqt_graph
```



Sprawdzono działanie programu w konsoli. Gdy sterowano żółciem przy pomocy węzła `/turtle1/cmd_vel` utworzony przez nas węzeł wyświetlał w konsoli, jakie wartości przyjmują odpowiednie prędkości nadane żółtowi.

```
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ source ~/ServiceRobotsLab/devel/setup.sh
andrzej@andrzej-VirtualBox:~/ServiceRobotsLab$ rosrn vrep_motor_controller vrep_motor_controller_node _
velocity_input:=/turtle1/cmd_vel
Hello world x
Velocity topic: /turtle1/cmd_vel
Node running
Node running
Node running
Node running
Node running
Node running
Node running
Node running
Node running
Velocity: Linear = 0, Angular = 2
Node running
Velocity: Linear = 2, Angular = 0
Node running
Velocity: Linear = 2, Angular = 0
Node running
Node running
Velocity: Linear = 0, Angular = 2
Node running
Velocity: Linear = 2, Angular = 0
Node running
Node running
Node running
Node running
Node running
Node running
Node running
```

Zad 2.

Zaktualizowano wcześniej napisany kod.

```
#include "ros/ros.h"
#include "std_msgs/Float64.h"
#include "geometry_msgs/Twist.h"

void twist_callback(const geometry_msgs::TwistConstPtr &twist_msg){
    double lin_vel = twist_msg->linear.x;
    double ang_vel = twist_msg->angular.z;
    std::cout << "Velocity: Linear = " << lin_vel << ", Angular = " << ang_vel << std::endl;
}

int main(int argc, char **argv) {
    ros::init(argc, argv, "vrep_motor_controller_node");
    ros::NodeHandle node_handle("~");
    std::cout << "Hello world x" << std::endl;
    ros::Rate loop_rate(1);
```

```

std::string velocity_cmd_topic;

node_handle.param<std::string>("velocity_input", velocity_cmd_topic, "cmd_vel");

// Wheather proper topic is specified or not
if(node_handle.getParam("velocity_input", velocity_cmd_topic))
{
    std::cout << "Velocity topic: "<< velocity_cmd_topic << std::endl;
} else
{
    std::cout << "Use default value" << std::endl;
}

// Subscriber object declaration
ros::Subscriber sub = node_handle.subscribe(velocity_cmd_topic, 1, twist_callback);

// Publisher object declaration
ros::Publisher pub = node_handle.advertise<std_msgs::Float64>("my_topic" , 1 );

// Temporary value of velocities
std_msgs::Float64 lin_vel;
lin_vel.data = 2.5;
std_msgs::Float64 ang_vel;
ang_vel.data = 1.8;
std_msgs::Float64 wheel_r;
wheel_r.data = 2.0;

// Data to being published
std_msgs::Float64 data_to_publish;
data_to_publish.data = 20*lin_vel.data/wheel_r.data - ang_vel.data*1;
pub.publish(data_to_publish);

// main loop
while (ros::ok() ) {

    ros::spinOnce();
    std::cout << "Node running" << std::endl;
    pub.publish(data_to_publish);
    loop_rate.sleep();
}
}

```

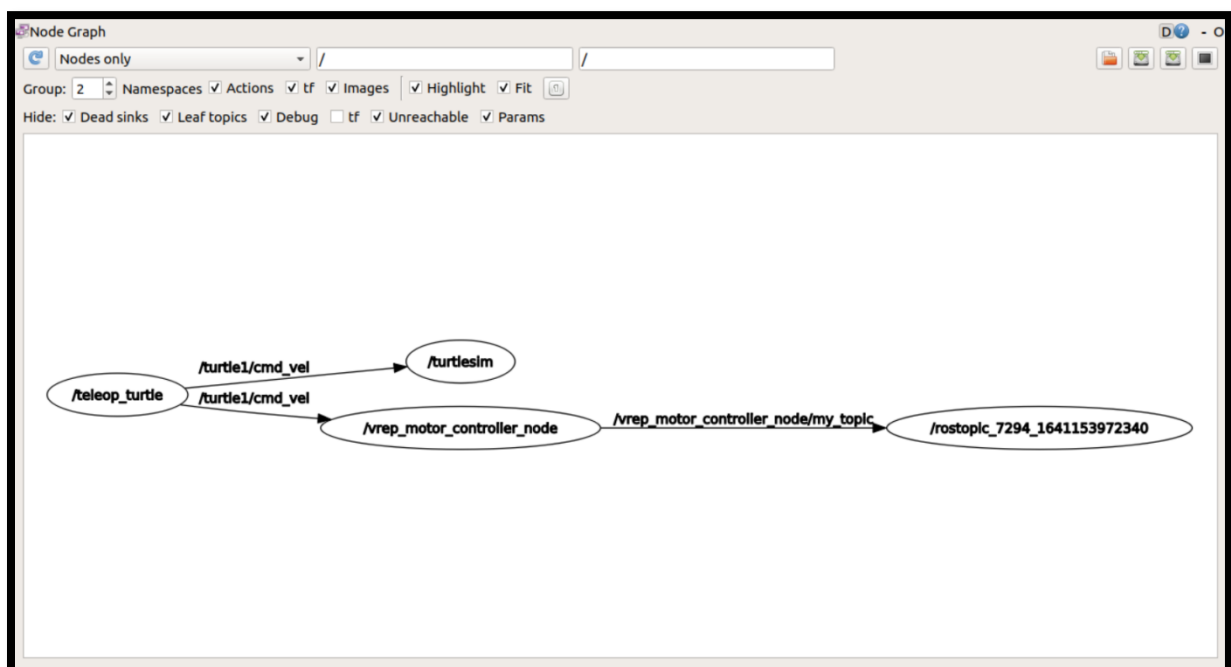
Dodano linijki tworzące nowy temat *my_topic*, który publikuje wartość wyliczoną w zmiennej *data_to_publish.data*. Roboczo ustawiono wartości *lin_vel* oraz *ang_vel* na stałe.

Publikowane przez węzeł dane:

```
andrzej@andrzej-VirtualBox: ~
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
andrzej@andrzej-VirtualBox:~$ rostopic echo /vrep_motor_controller_node/my_topic

data: 23.2
---
data: 23.2
---
data: 23.2
---
data: 23.2
---
data: 23.2
---
data: 23.2
---
data: 23.2
---
data: 23.2
---
data: 23.2
---
```

Graf rqt:



Temat do którego publikowane są dane jest widoczny po prawej stronie grafu.

Zad3.

Zmodyfikowano dotychczas napisany kod. W ramach jednego węzła zaimplementowano generowanie liczb losowych, wyznaczanie na ich podstawie wartości prędkości żółwia, oraz publikowanie wiadomości do tematu `/turtle1/cmd_vel`.

Napisany kod:

```
#include "ros/ros.h"
#include "std_msgs/Float64.h"
#include "geometry_msgs/Twist.h"

#include <iostream>
#include <cstdlib>
#include <ctime>

#define MIN_VAL -10.0
#define MAX_VAL 10.0

// Function generating random float number
float get_random_num(float min, float max){
    srand((unsigned int)time(NULL));
    return (float(rand()) / float((RAND_MAX)) * 2*max) + min;
}

int main(int argc, char **argv) {
    ros::init(argc, argv, "vrep_motor_controller_node");
    ros::NodeHandle node_handle("~");
    std::cout << "Hello world x" << std::endl;
    ros::Rate loop_rate(2);

    // Publisher object declaration
    ros::Publisher pub = node_handle.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel", 1);

    // Data to being published
    geometry_msgs::Twist vel_msg;

    float divide_by = get_random_num(-2.0, 5.0);

    // main loop
    while (ros::ok() ) {

        ros::spinOnce();
```

```

// Generating new random values to being published
vel_msg.linear.x = get_random_num(MIN_VAL,MAX_VAL) /1.77;
vel_msg.angular.z = get_random_num(MIN_VAL, MAX_VAL) / divide_by;

// Publishing velocity message (Twist type)
pub.publish(vel_msg);

std::cout << "Linear x: "<< vel_msg.linear.x << std::endl;
std::cout << "Angular z: "<< vel_msg.angular.z << std::endl;

loop_rate.sleep();
divide_by = get_random_num(-2.0, 5.0);
}
}

```

Dodano nowe biblioteki potrzebne do implementacji generatora liczb pseudolosowych. Zdefiniowano globalne wartości MIN_VAL = -10.0 oraz MAX_VAL = 10.0 są to progi zakresu z którego ma zostać wylosowana liczba.

Dalej zadeklarowano funkcję zwracającą wygenerowaną liczbę pseudolosową typu *float*. Kolejno zapisano deklarację obiektu typu *Publisher* oraz zmienną typu *Twist* która będzie publikowana.

W pętli *main* prędkości są obliczane z prostej formuły:

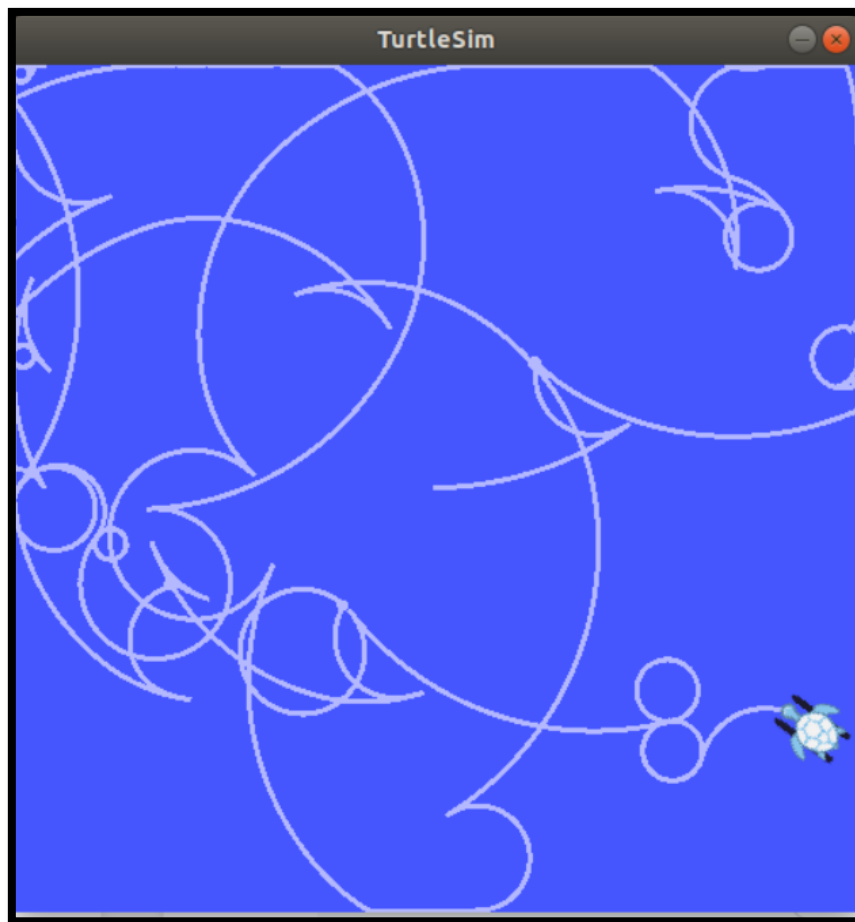
```

// Generating new random values to being published
vel_msg.linear.x = get_random_num(MIN_VAL,MAX_VAL) /1.77;
vel_msg.angular.z = get_random_num(MIN_VAL, MAX_VAL) / divide_by;

```

Została ona zastosowana w ramach testów.

Przykładowa trasa, którą pokonał żółwik:



Generowane prędkości, wypisane w konsoli:

```
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
Angular z: 8.84796
P Linear x: 5.35397
1 Angular z: 1.22463
[ Linear x: 5.35397
8 Angular z: 1.22463
[ Linear x: -1.96662
4 Angular z: -2.76362
[ Linear x: -1.96662
3 Angular z: -2.76362
[ Linear x: 2.11286
9 Angular z: 0.767938
[ Linear x: 2.11286
7 Angular z: 0.767938
[ Linear x: -5.14006
0 Angular z: 5.87357
[ Linear x: -5.14006
4 Angular z: 5.87357
[ Linear x: -1.04039
2 Angular z: -0.88565
[ Linear x: -1.04039
7 Angular z: -0.88565
[ Linear x: -2.69753
9 Angular z: -7.793
[
```

Graf rqt:

