Tadeusz Chmielik Michał Pióro

Laboratorium 4 STERO

Grudzień 2024

1. Laboratorium

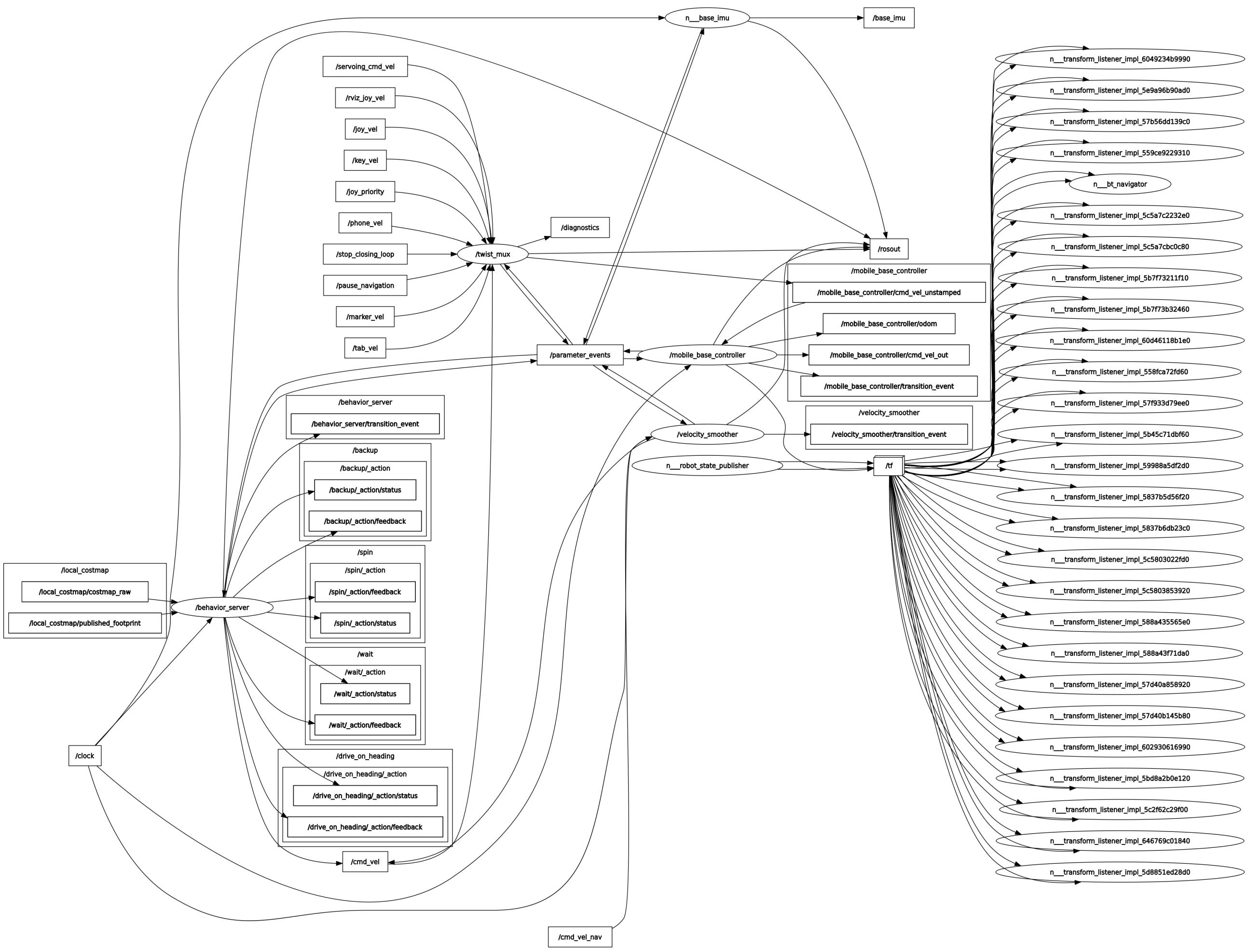
1.1. Analiza systemu robota TIAGo

System pozwalający na symulację systemu robota uruchomiono z sukcesem co pozwoliło na jego przeanalizowanie oraz stworzenie pierwszego węzła.

1.1.1. struktura sterowania

prędkość bazy

W celu zadania prędkości bazy należy skorzystać z jednego z dwóch topiców /cmd_vel, /cmd_nav (pierwszy najczęściej do ręczenego sterowania przez użytkownika, drugi do autopmatycznej nawigacji), które udostępniane są przez node moblie_base_controller. W celu zidentyfikowania tego skorzystano z rqt_graph-a.



Rys. 1.1. rqt_graph dla węzła moblie_base_controller

```
    student@gepard:~/stero$ ros2 topic info /cmd_vel
Type: geometry_msgs/msg/Twist
Publisher count: 5
Subscription count: 1
student@gepard:~/stero$ ros2 topic info /cmd_vel_nav
Type: geometry_msgs/msg/Twist
Publisher count: 1
Subscription count: 1
```

Odometria

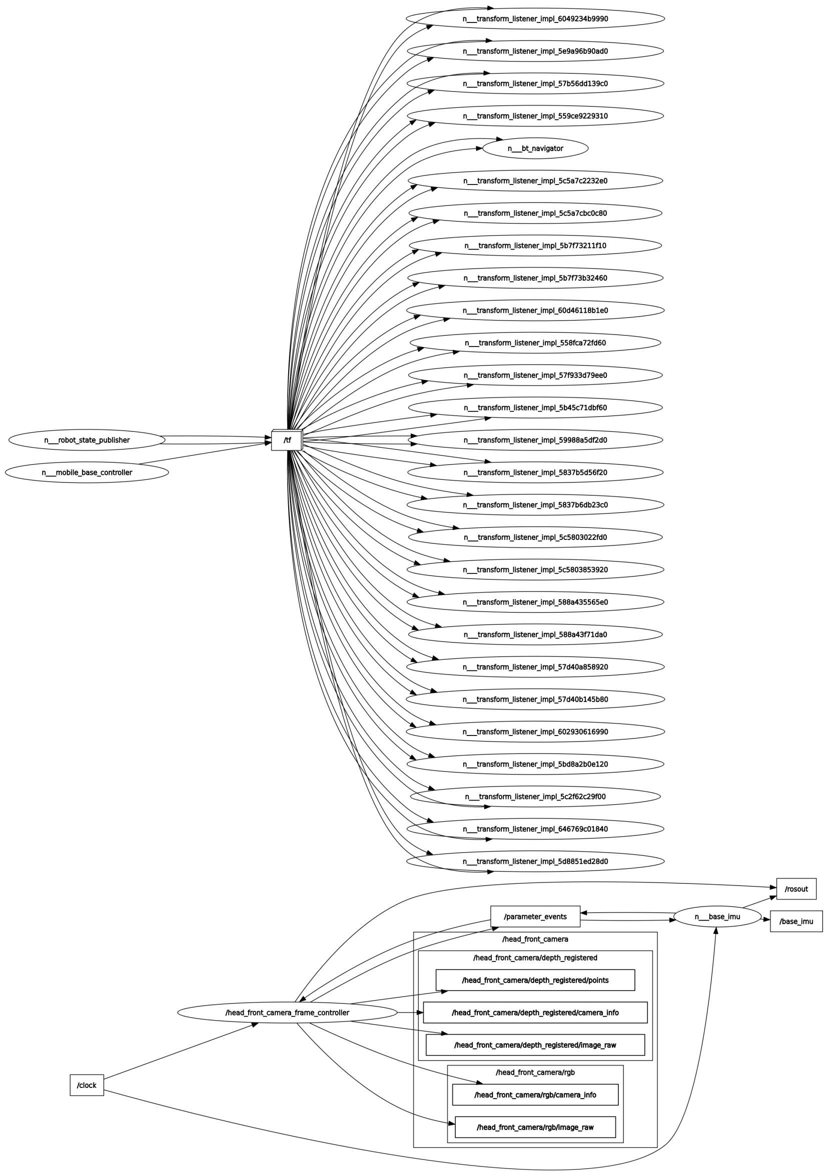
W celu odczytu odometrii robota można wykorzystać topic /mobile_base_controler/odom który można znaleźć na wyżej pokazanym grafie.

```
ros2 topic info /mobile_base_controller/odom
Type: nav_msgs/msg/Odometry
Publisher count: 1
Subscription count: 3
```

Czujnik LiDAR

Za obsługę czujnika LiDAR odpowiada węzeł /base_laser. Ponadto do publikowania danych wykorzystywane są tematy /scan oraz /scan_raw. Dzięki serwisom: /base_laser, /describe_parameters, /base_laser, /get_parameter_types, /base_laser, /get_parameters, /base_laser, /get_type_descripti /base_laser/, list_parameters /base_laser, /set_parameters, /base_laser, /set_parameters_atomically możliwe jest konfigurowanie LiDARa.

kamera RGB-D



Rys. 1.2. rqt_graph dla węzła head_front_camera_controller

Jak można zauważyć do obsługi kamery oraz do wysyłania danych przez nią zebranych wykorzystano wiele węzłów udostępniających wiele danych na wiele sposobów.

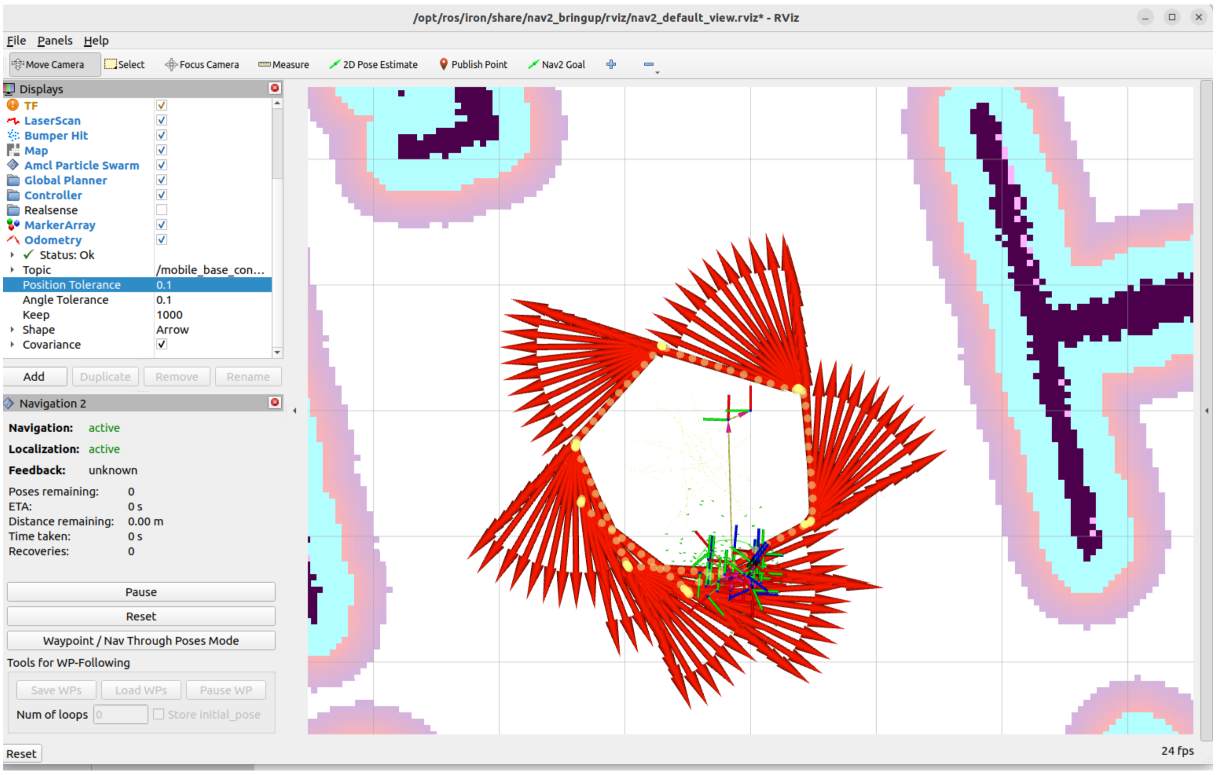
```
    ros2 node info /head_front_camera_frame_controller
/head_front_camera_frame_controller
  Subscribers:
    /clock: rosgraph_msgs/msg/Clock
    /parameter_events: rcl_interfaces/msg/ParameterEvent
  Publishers:
    /head_front_camera/depth_registered/camera_info: sensor_msgs/msg/CameraInfo
    /head_front_camera/depth_registered/image_raw: sensor_msgs/msg/Image
    /head_front_camera/depth_registered/points: sensor_msgs/msg/PointCloud2
    /head_front_camera/rgb/camera_info: sensor_msgs/msg/CameraInfo
    /head_front_camera/rgb/image_raw: sensor_msgs/msg/Image
```

```
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /head_front_camera_frame_controller/describe_parameters: rcl_interfaces/srv/
    /head_front_camera_frame_controller/get_parameter_types: rcl_interfaces/srv/
    /head_front_camera_frame_controller/get_parameters: rcl_interfaces/srv/GetPa
    /head_front_camera_frame_controller/get_type_description: type_description_i
    /head_front_camera_frame_controller/list_parameters: rcl_interfaces/srv/ListP
    /head_front_camera_frame_controller/set_camera_info: sensor_msgs/srv/SetCame
    /head_front_camera_frame_controller/set_parameters: rcl_interfaces/srv/SetPa
    /head_front_camera_frame_controller/set_parameters_atomically: rcl_interfaces
  Service Clients:

  Action Servers:

  Action Clients:
```

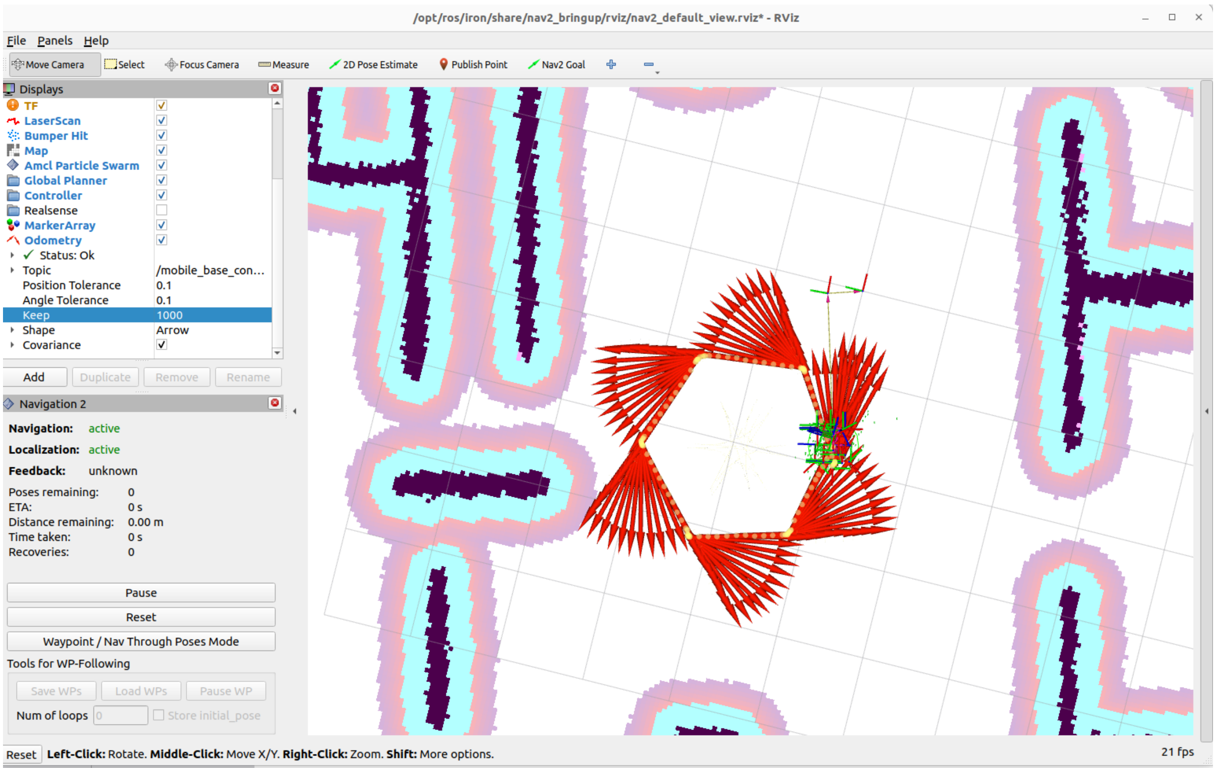
1.2. Węzeł test_nav

1.2.1. Opis wykonanego zadania

W ramach zadania napisano w języku C++ węzeł *test_nav*, którego zadaniem było sterowanie robotem Tiago w taki sposób, aby poruszał się w środpowisku symulacyjnym po trasie w kształcie sześciokąta. Aktualna prędkość liniowa i kątowa jest publikowana na temacie */cmd_vel*. Informacje o aktualnym położeniu i orientacji robota jest pobierana z tematu */odom*, na tej podstawie po przejechaniu zadanej odległości lub obrocie o kąt 120 stopni następowała zmiana publikowanych prędkości. Początkowo przy większej prędkości robot miał problem z dokładnym przejechaniem wyznaczonej trasy, może to być spowodowane opóźnieniami w komunikacji wewnątrz systemu, a także z bezwładności symulowanego robota.



Rys. 1.3. Wizualizacja trasy przejazdu i kolejnych wektorów położenia robota przy zbyt dużej prędkości



Rys. 1.4. Wizualizacja trasy przejazdu i kolejnych wektorów położenia robota dla optymalnych parametrów

1.2.2. Kod

Węzeł C++

```
    #include <rclcpp/rclcpp.hpp>
#include <nav_msgs/msg/odometry.hpp>
#include <geometry_msgs/msg/twist.hpp>
#include <geometry_msgs/msg/pose.hpp>
#include <cmath>
#include <chrono>

using namespace std::chrono_literals;

// Definicja krawędzi sześciokąta i prędkości robota
const double SIDE_LENGTH = 1.0;  // Długość boku sześciokąta (w metrach)
const double LINEAR_VELOCITY = 0.2;  // Prędkość liniowa (w metrach na sekundę)
const double ANGULAR_VELOCITY = 3.14 / 18.0;  // Prędkość kątowa (60 stopni na se
const double ANGULAR_ROTATION = 3.14 / 3.0;

class OdomListener : public rclcpp::Node
{
public:
    OdomListener() : Node("odom_listener")
    {
        subscription_ = this->create_subscription<nav_msgs::msg::Odometry>(
            "/mobile_base_controller/odom", 10, std::bind(&OdomListener::odomCal

        cmd_vel_pub_ = this->create_publisher<geometry_msgs::msg::Twist>("/cmd_v

        current_side_ = 0;
        move_forward_ = true;

        prev_position_ = geometry_msgs::msg::Pose();
        prev_orientation_ = 0.0; // Inicjalizacja kąta obrotu
    }

private:
    rclcpp::Subscription<nav_msgs::msg::Odometry>::SharedPtr subscription_;
    rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr cmd_vel_pub_;

    int current_side_;  // Numer aktualnego boku sześciokąta
    bool move_forward_;  // Flaga wskazująca, czy robot jedzie do przodu, czy sk
    geometry_msgs::msg::Pose prev_position_;  // Poprzednia pozycja robota
    double prev_orientation_;  // Poprzednia orientacja robota (kąt)

    void odomCallback(const nav_msgs::msg::Odometry::SharedPtr msg)
    {
        double current_x = msg->pose.pose.position.x;
        double current_y = msg->pose.pose.position.y;

        double dx = current_x - prev_position_.position.x;
        double dy = current_y - prev_position_.position.y;
```

```
        double distance_travelled = std::sqrt(dx * dx + dy * dy);

        double current_orientation = getYaw(msg->pose.pose.orientation);

        double delta_orientation = current_orientation - prev_orientation_;

        if (delta_orientation > 3.14)
        {
            delta_orientation -= 2 * 3.14;
        }
        else if (delta_orientation < -3.14)
        {
            delta_orientation += 2 * 3.14;
        }

        if (move_forward_)
        {
            if (distance_travelled >= SIDE_LENGTH)
            {
                move_forward_ = false;
                prev_position_ = msg->pose.pose;
                prev_orientation_ = current_orientation;
            }
        }
        else
        {
            if (std::abs(delta_orientation) >= ANGULAR_ROTATION)
            {
                // Po obróceniu o zadany kąt, robot kontynuuje jazdę do przodu
                move_forward_ = true;
                prev_position_ = msg->pose.pose;
                prev_orientation_ = current_orientation;
                current_side_++;
            }
        }

        moveInHexagon();

        RCLCPP_INFO(this->get_logger(), "Current Angle: %f, Distance travelled: %
    }

    double getYaw(const geometry_msgs::msg::Quaternion& quat)
    {
        // Wzór na wyliczenie kąta yaw z kwaternionu
        double siny_cosp = 2.0 * (quat.w * quat.z + quat.x * quat.y);
        double cosy_cosp = 1.0 - 2.0 * (quat.y * quat.y + quat.z * quat.z);
        return std::atan2(siny_cosp, cosy_cosp);
    }

    void moveInHexagon()
    {
```

```
        auto msg = geometry_msgs::msg::Twist();

        if (move_forward_)
        {
            // Poruszamy się do przodu
            msg.linear.x = LINEAR_VELOCITY;
            msg.angular.z = 0.0;
        }
        else
        {
            // Skręcamy o 60 stopni
            msg.linear.x = 0.0;
            msg.angular.z = ANGULAR_VELOCITY;
        }

        // Publikacja prędkości
        cmd_vel_pub_->publish(msg);
    }
};

int main(int argc, char * argv[])
{
    rclcpp::init(argc, argv);
    rclcpp::spin(std::make_shared<OdomListener>());
    rclcpp::shutdown();
    return 0;
}
```

CMake

```
cmake_minimum_required(VERSION 3.8)
project(lab4)

if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(nav_msgs REQUIRED)

add_executable(test_nav src/test_nav.cpp)

target_include_directories(test_nav PUBLIC
  $<BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include>
  $<INSTALL_INTERFACE:include>)

target_compile_features(test_nav PUBLIC c_std_99 cxx_std_17)

ament_target_dependencies(test_nav rclcpp nav_msgs)
```

```
install(TARGETS test_nav
  DESTINATION lib/${PROJECT_NAME})

if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  set(ament_cmake_copyright_FOUND TRUE)
  set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()

ament_package()
```

2. Projekt

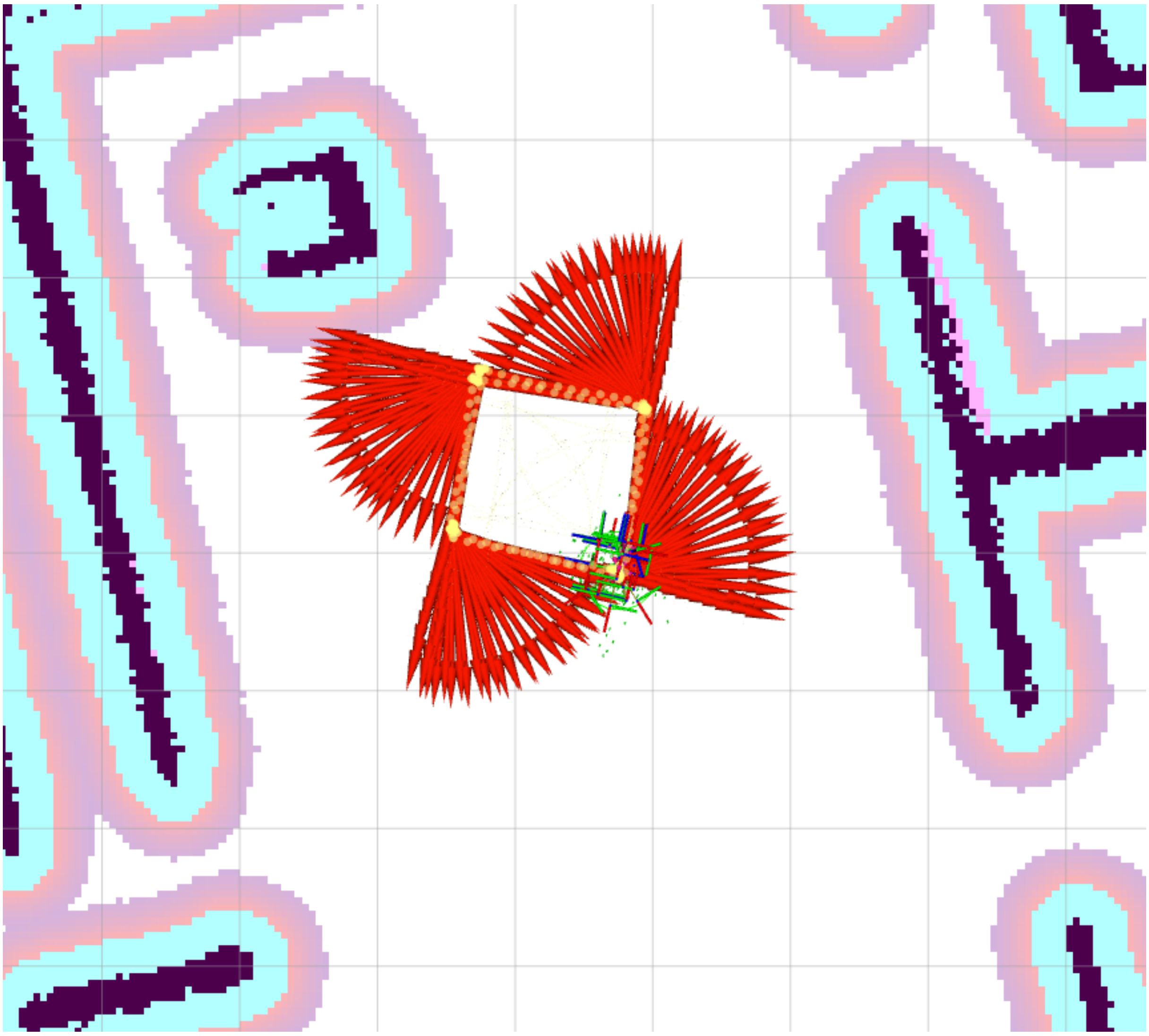
2.1. Nawigacja robota po kwadracie

2.1.1. Opis

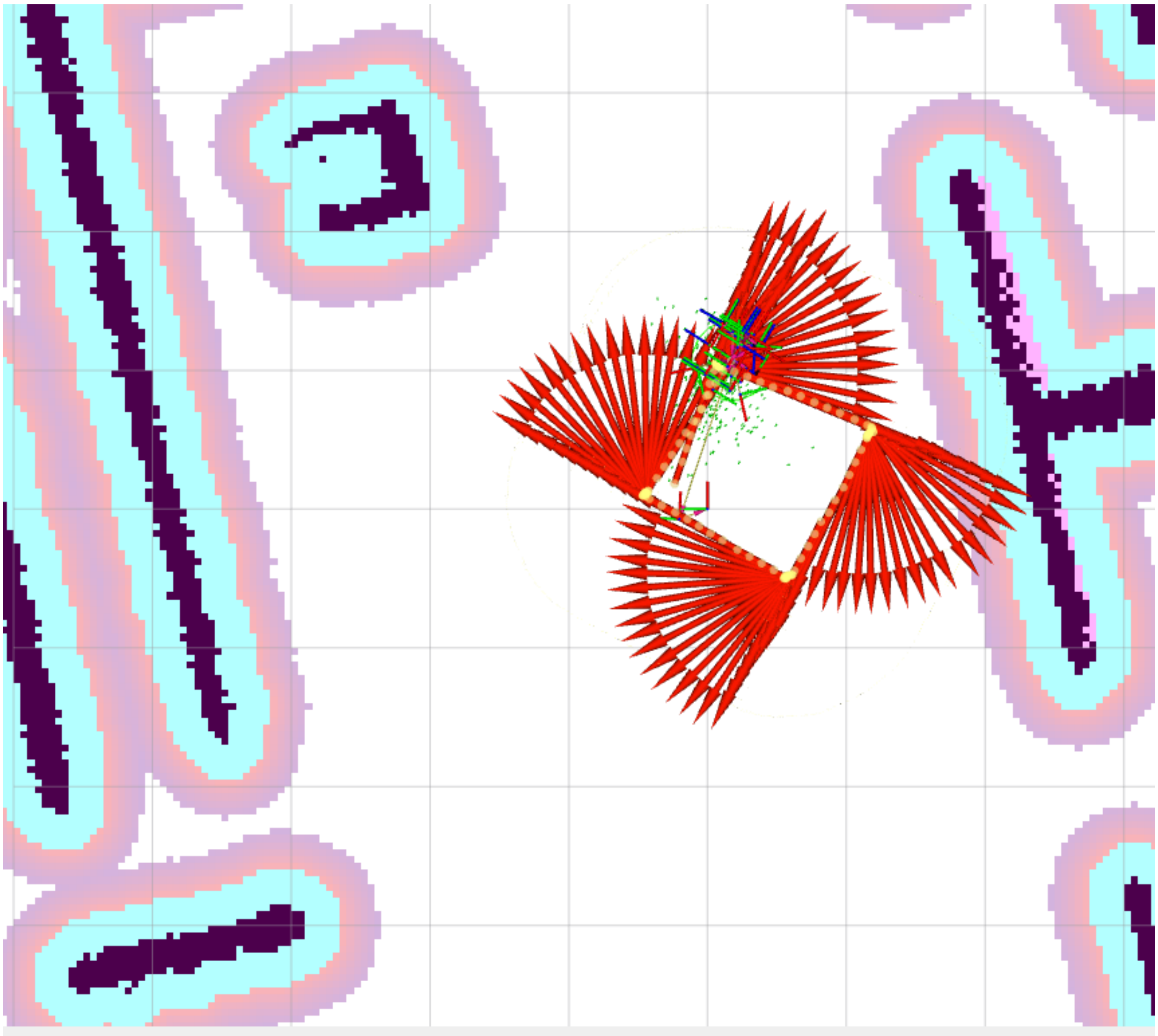
W ramach zadania projektowegon stworzono węzeł sterujący ruchem robota po trasie w kształcie kwadratu. Podobnie jak na laboratorium wykorzystano temat */cmd_vel* do zadawania prędkości robota oraz temat */mobile_base_controller/odom* do bieżącej weryfikacji położenia robota i na tej podstawie decydowanie o zadawanej prędkości. Jako argument wywołania programu *square_nav* należy podać długość boku kwadratu, liczbę okrążeń oraz ich kierunek. Przykładowe wywołanie programu: *ros2 run lab4 square_nav –ros-args -p side_length:=1.1 -p num_laps:=10 -p clockwise:=true*. Program co 10s symulacji zwraca informację o chwilowym błędzie średniokwadratowym, oraz po każdym okrązeniu wylicza średni błąd. Na koniec zwraca podsumowanie w tabeli oraz jest wyłączany.

2.1.2. Testy

W ramach przeprowadzonych testów zadano kwadrat o boku 1,1m okrążony 10 razy zgodnie i przeciwnie do ruchu wskazówek zegara. Zauważono, że wraz z upływem czasu symulacji rosną chwilowe błędy średniokwadratowe oraz średnie błędy pozycjonowania z danego okrążenia. Natomiast błąd orientacji bardziej oscyluje. Ponadto program lepiej radzie sobie z kierunkiem odwrotnym do ruchu wskazówek zegara. Może to wynikać z bezwładności robota i jego fizycznych uwarunkowań.



Rys. 2.1. Wizaulizacja trasy przejazdu zgodnie z ruchem wskazówek zegara



Rys. 2.2. Wizaulizacja trasy przejazdu odwrotnie do ruchu wskazówek zegara

Zwracana tabela błędów - kierunek zgodny z ruchem wskazówek zegara

```
==================== Temporary Errors ====================
    Time (s)      |   Position Error   |  Orientation Error
       10.015  |      0.000396516  |          0.000639386
      20.0369  |      0.000396499  |          0.000639394
      30.0457  |      0.000396499  |          0.000639384
      40.0491  |      0.000396489  |          0.000639402
      50.0542  |      0.000396485  |           0.00063939
      60.0733  |      0.000396477  |          0.000639421
       70.084  |      0.000396445  |          0.000639418
      80.0851  |      0.000396321  |          0.000640031
      90.0951  |      0.000395969  |          0.000640808
      100.108  |      0.000395038  |           0.00064265
      110.111  |      0.000394661  |          0.000641984
      120.115  |      0.000391946  |           0.00064476
      130.116  |      0.000392604  |          0.000642185
      140.134  |      0.000387915  |          0.000644983
      150.141  |      0.000390275  |          0.000642179
```

160.153	0.00038344	0.00064498
170.159	0.000387701	0.000642174
180.173	0.000378669	0.000644963
190.179	0.000384951	0.000642158
200.189	0.00037371	0.000644933
210.245	0.000382053	0.000642145
220.253	0.000368614	0.000644909
230.276	0.000379025	0.000642134
240.277	0.000363416	0.000644892
250.288	0.000375877	0.000642128
260.292	0.000358132	0.000644883
270.304	0.000372616	0.000642125
280.313	0.000352775	0.000644879
290.313	0.000369246	0.000642124
300.331	0.00034735	0.000644882
310.355	0.000365769	0.000642127
320.356	0.000341862	0.000644889
330.364	0.000362189	0.000642131
340.382	0.000336313	0.000644901
350.398	0.000358505	0.000642138
360.413	0.000330707	0.000644916
370.419	0.000354722	0.000642145
380.422	0.000325045	0.000644933
390.426	0.000350839	0.000642154
400.433	0.000319328	0.000644952
410.453	0.000346859	0.000642163
420.472	0.000313695	0.000644957
430.487	0.00034332	0.000642118
440.514	0.000309197	0.000644832
450.561	0.000340438	0.000642022
460.583	0.000305772	0.000644641
470.61	0.000337936	0.000641929
480.619	0.00030296	0.000644463
490.63	0.000335656	0.000641851
500.651	0.000300503	0.000644316
510.66	0.000333509	0.000641789
520.676	0.000298253	0.000644202
530.683	0.000331442	0.000641742
540.69	0.000296127	0.000644115
550.691	0.000329426	0.000641707
560.695	0.000294076	0.000644051
570.735	0.000327445	0.000641681
580.741	0.000292074	0.000644005
590.746	0.000325487	0.000641662
600.767	0.000290104	0.00064397
610.777	0.000323548	0.000641648
620.781	0.000288158	0.000643946
630.785	0.000321625	0.000641638
640.8	0.00028623	0.000643928
650.83	0.000319716	0.000641631
660.835	0.000284317	0.000643916

670.849	0.000317819	0.000641625
680.858	0.000282418	0.000643907
690.873	0.000315934	0.000641621
700.875	0.000280531	0.000643901
710.88	0.000314061	0.000641618
720.884	0.000278656	0.000643897
730.885	0.000312199	0.000641616
740.908	0.000276793	0.000643895
750.937	0.000310349	0.000641615
760.947	0.000274942	0.000643893
770.98	0.000308511	0.000641613
781.001	0.000273102	0.000643892
791.01	0.000306683	0.000641612
801.016	0.000271274	0.000643892
811.028	0.000304867	0.000641612
821.041	0.000269456	0.000643892
831.082	0.000303062	0.000641611
841.083	0.000267651	0.000643892
851.096	0.000301268	0.00064161
861.106	0.000265856	0.000643892
871.129	0.000299486	0.00064161
881.134	0.000264073	0.000643893
891.141	0.000297715	0.000641609
901.145	0.000262301	0.000643893
911.146	0.000295955	0.000641609
921.166	0.000260541	0.000643893
931.177	0.000294206	0.000641608
941.182	0.000258791	0.000643894
951.199	0.000292469	0.000641608
961.201	0.000257053	0.000643895
971.201	0.000290743	0.000641607
981.225	0.000255327	0.000643895
991.234	0.000289028	0.000641607
1001.24	0.000253612	0.000643896
1011.25	0.000287324	0.000641606
1021.27	0.000251908	0.000643896
1031.29	0.000285632	0.000641606
1041.3	0.000250215	0.000643897
1051.32	0.000283951	0.000641605
1061.33	0.000248534	0.000643897
1071.35	0.000282282	0.000641605
1081.35	0.000246864	0.000643897
1091.36	0.000280623	0.000641604
1101.37	0.000245205	0.000643898
1111.38	0.000278976	0.000641604
1121.4	0.000243558	0.000643898
1131.4	0.00027734	0.000641603
1141.42	0.000241922	0.000643898
1151.44	0.000275716	0.000641602
1161.45	0.000240297	0.000643898
1171.48	0.000274103	0.000641602

```
   1181.49 |     0.000238684 |        0.000643898
   1191.49 |     0.000272501 |        0.000641601
   1201.49 |     0.000237082 |        0.000643898
    1211.5 |      0.00027091 |          0.0006416
   1221.56 |     0.000235491 |        0.000643898
   1231.57 |     0.000269331 |        0.000641599
   1241.58 |     0.000233912 |        0.000643898
   1251.59 |     0.000267763 |        0.000641598
    1261.6 |     0.000232344 |        0.000643897
    1271.6 |     0.000266206 |        0.000641597
   1281.61 |     0.000304426 |        0.000639301
   1291.63 |      0.00026466 |        0.000641596
   1301.65 |     0.000229242 |        0.000643896
   1311.67 |     0.000263126 |        0.000641596
   1321.67 |     0.000227708 |        0.000643896
   1331.68 |     0.000261603 |        0.000641595
   1341.69 |     0.000226185 |        0.000643896
    1351.7 |     0.000260092 |        0.000641594
   1361.74 |     0.000224674 |        0.000643895
   1371.74 |     0.000258592 |        0.000641593
   1381.75 |     0.000223174 |        0.000643895
   1391.76 |     0.000257103 |        0.000641592
   1401.78 |     0.000221685 |        0.000643894
   1411.79 |     0.000255625 |         0.00064159
   1421.79 |     0.000220208 |        0.000643893
   1431.81 |     0.000254159 |        0.000641589
   1441.81 |     0.000218742 |        0.000643893
   1451.81 |     0.000252704 |        0.000641588
   1461.83 |     0.000217287 |        0.000643892
   1471.84 |      0.00025126 |        0.000641587
   1481.84 |     0.000215844 |        0.000643891
   1491.84 |     0.000249828 |        0.000641586
   1501.85 |     0.000214412 |         0.00064389
   1511.86 |     0.000248407 |        0.000641585
   1521.87 |     0.000212991 |         0.00064389
   1531.88 |     0.000246997 |        0.000641584
   1541.89 |     0.000211582 |        0.000643889
   1551.92 |     0.000245598 |        0.000641583
   1561.92 |     0.000210184 |        0.000643888
   1571.94 |     0.000244211 |        0.000641581

=============== Cumulative Errors ===============
   Lap    | Avg Position Error | Avg Orientation Error
        1 |         0.00113027 |              0.106544
        2 |         0.00111898 |              0.096678
        3 |         0.00110479 |             0.0864804
        4 |         0.00109004 |             0.0763321
        5 |         0.00107421 |             0.0712465
        6 |         0.00105804 |             0.0712182
        7 |         0.00104198 |             0.0559962
        8 |          0.0010263 |             0.0509008
```

```
      9 |        0.00101127 |              0.0508849
     10 |       0.000997105 |                0.03564
```

Zwracana tabela błędów - kierunek odwrotny do ruchu wskazówek zegara

```
================ Temporary Errors ================
   Time (s)     |  Position Error  | Orientation Error
     10.0105 |       0.0154914 |          0.0431744
     20.0171 |       0.0154909 |          0.0431744
     30.0227 |       0.0154904 |          0.0431744
     40.0337 |       0.0154901 |          0.0431744
     50.0427 |         0.01549 |          0.0431744
     60.0802 |     1.76971e-05 |        3.03218e-05
     70.0934 |     1.76919e-05 |        3.03236e-05
     80.1033 |     1.76895e-05 |        3.03208e-05
     90.1167 |     1.76862e-05 |        3.03232e-05
     100.118 |     1.76844e-05 |        3.03198e-05
     110.145 |      1.7682e-05 |         3.0323e-05
     120.178 |     1.76807e-05 |          3.032e-05
      130.18 |      1.7679e-05 |        3.03246e-05
     140.203 |     1.76781e-05 |        3.03221e-05
     150.207 |     1.76768e-05 |        3.03278e-05
     160.212 |     1.76758e-05 |         3.0323e-05
     170.215 |     1.76747e-05 |         3.0328e-05
     180.231 |      1.7674e-05 |        3.03222e-05
     190.235 |     1.76732e-05 |        3.03271e-05
     200.246 |     1.76726e-05 |        3.03223e-05
     210.252 |     1.76719e-05 |        3.03266e-05
      220.26 |     1.76714e-05 |         3.0322e-05
     230.278 |     1.76709e-05 |        3.03254e-05
     240.293 |     1.76705e-05 |        3.03216e-05
     250.297 |     1.76701e-05 |        3.03237e-05
       260.3 |     1.76699e-05 |        3.03211e-05
     270.309 |     1.76698e-05 |        3.03218e-05
      280.31 |     1.76697e-05 |        3.03208e-05
     290.313 |     1.76697e-05 |        3.03208e-05
     300.317 |     1.76697e-05 |        3.03223e-05
     310.321 |     1.76697e-05 |        3.03223e-05
     320.325 |     1.76696e-05 |         3.0323e-05
      330.34 |     1.76696e-05 |         3.0323e-05
     340.341 |     1.76696e-05 |        3.03232e-05
     350.352 |     1.76696e-05 |        3.03232e-05
     360.357 |     1.76695e-05 |        3.03233e-05
     370.358 |     1.76695e-05 |        3.03233e-05
      380.38 |     1.76694e-05 |        3.03234e-05
     390.389 |     1.76694e-05 |        3.03235e-05
     400.397 |     1.76694e-05 |        3.03236e-05
       410.4 |     1.76694e-05 |        3.03236e-05
     420.419 |     1.76693e-05 |        3.03238e-05
     430.426 |     1.76693e-05 |        3.03238e-05
     440.427 |     1.76693e-05 |        3.03241e-05
```

450.439	1.76693e−05	3.03241e−05
460.449	1.76692e−05	3.03243e−05
470.45	1.76692e−05	3.03244e−05
480.463	1.76692e−05	3.03241e−05
490.466	1.76707e−05	3.02611e−05
500.473	1.76177e−05	3.02425e−05
510.475	1.74562e−05	3.02241e−05
520.484	1.74441e−05	3.00096e−05
530.485	1.68989e−05	2.96254e−05
540.487	1.70194e−05	2.99473e−05
550.5	1.61511e−05	2.96033e−05
560.501	1.66077e−05	3.00439e−05
570.512	1.53962e−05	2.98449e−05
580.522	1.61738e−05	3.02076e−05
590.524	1.46469e−05	3.01723e−05
600.529	1.574e−05	3.03687e−05
610.533	1.39301e−05	3.04884e−05
620.538	1.53086e−05	3.05136e−05
630.554	1.32546e−05	3.07676e−05
640.569	1.48832e−05	3.06349e−05
650.583	1.2627e−05	3.09998e−05
660.595	1.44654e−05	3.07331e−05
670.606	1.74902e−05	3.0323e−05
680.615	1.40559e−05	3.08109e−05
690.618	1.15258e−05	3.13356e−05
700.625	1.36552e−05	3.08718e−05
710.638	1.1055e−05	3.14517e−05
720.639	1.32636e−05	3.09191e−05
730.657	1.0638e−05	3.15419e−05
740.662	1.28813e−05	3.09557e−05
750.67	1.02752e−05	3.1612e−05
760.677	1.25084e−05	3.09841e−05
770.684	9.96671e−06	3.16665e−05
780.689	1.21449e−05	3.10061e−05
790.711	9.7127e−06	3.17089e−05
800.754	1.17909e−05	3.10232e−05
810.757	9.51322e−06	3.17422e−05
820.76	1.14464e−05	3.10367e−05
830.778	9.36843e−06	3.17648e−05
840.787	1.11677e−05	3.10349e−05
850.8	9.26789e−06	3.17526e−05
860.816	1.09692e−05	3.10199e−05
870.826	9.18012e−06	3.17216e−05
880.828	1.08142e−05	3.10038e−05
890.833	9.09149e−06	3.16903e−05
900.837	1.06837e−05	3.09897e−05
910.845	9.00009e−06	3.16637e−05
920.856	1.05671e−05	3.09784e−05
930.863	8.90642e−06	3.16427e−05
940.869	1.04588e−05	3.09697e−05
950.882	8.81139e−06	3.16268e−05

```
     960.888 |       1.03553e-05 |         3.09631e-05
     970.889 |       8.71573e-06 |          3.1615e-05
     980.904 |       1.02548e-05 |         3.09582e-05
     990.914 |       8.61989e-06 |         3.16062e-05
     1000.92 |       1.01562e-05 |         3.09545e-05
     1010.94 |       8.52417e-06 |         3.15998e-05
     1020.96 |       1.00591e-05 |         3.09518e-05
     1030.96 |       8.42874e-06 |         3.15951e-05
     1040.97 |       9.96288e-06 |         3.09498e-05
     1050.98 |       8.33371e-06 |         3.15917e-05
     1060.99 |       9.86751e-06 |         3.09482e-05
        1071 |       8.23914e-06 |         3.15892e-05
     1081.01 |       9.77283e-06 |         3.09469e-05
     1091.01 |       8.14505e-06 |         3.15873e-05
     1101.01 |       9.67877e-06 |         3.09459e-05
     1111.01 |       8.05149e-06 |         3.15859e-05
     1121.03 |       9.58531e-06 |         3.09451e-05
     1131.04 |       7.95844e-06 |         3.15849e-05
     1141.06 |       9.49241e-06 |         3.09444e-05
     1151.07 |       1.53719e-05 |         3.03108e-05
     1161.08 |       9.40008e-06 |         3.09438e-05
     1171.09 |       7.77396e-06 |         3.15834e-05
     1181.09 |       9.30829e-06 |         3.09433e-05
      1191.1 |       7.68252e-06 |         3.15829e-05
      1201.1 |       9.21705e-06 |         3.09429e-05
     1211.12 |       7.59163e-06 |         3.15825e-05
     1221.13 |       9.12636e-06 |         3.09425e-05
     1231.13 |       7.50127e-06 |         3.15821e-05
     1241.15 |       9.03621e-06 |         3.09421e-05
     1251.15 |       7.41146e-06 |         3.15818e-05
     1261.16 |        8.9466e-06 |         3.09417e-05

=============== Cumulative Errors ===============
   Lap    | Avg Position Error | Avg Orientation Error
        1 |        3.99486e-05 |              0.020742
        2 |        4.22573e-05 |              0.026176
        3 |         4.9393e-05 |              0.036635
        4 |        5.88905e-05 |             0.0418887
        5 |        7.02383e-05 |             0.0523504
        6 |        8.32727e-05 |             0.0523729
        7 |        9.79444e-05 |             0.0628261
        8 |        0.000114207 |             0.0732737
        9 |        0.000132053 |             0.0785106
       10 |         0.00015148 |             0.0837508
```

2.1.3. Kod

Węzeł C++

```
#include <rclcpp/rclcpp.hpp>
#include <nav_msgs/msg/odometry.hpp>
```

```
#include <geometry_msgs/msg/twist.hpp>
#include <geometry_msgs/msg/pose.hpp>
#include <cmath>
#include <chrono>
#include <vector>
#include <numeric>

using namespace std::chrono_literals;

class SquareNav : public rclcpp::Node
{
public:

    SquareNav() : Node("square_nav")
    {
        this->declare_parameter<double>("side_length", 1.0);
        this->declare_parameter<int>("num_laps", 1);
        this->declare_parameter<bool>("clockwise", false);

        side_length_ = this->get_parameter("side_length").as_double();
        num_laps_ = this->get_parameter("num_laps").as_int();
        clockwise_ = this->get_parameter("clockwise").as_bool();

        odom_sub_ = this->create_subscription<nav_msgs::msg::Odometry>(
            "/mobile_base_controller/odom", 10, std::bind(&SquareNav::odomCallba
        ground_truth_sub_ = this->create_subscription<nav_msgs::msg::Odometry>(
            "/ground_truth_odom", 10, std::bind(&SquareNav::groundTruthCallback,

        cmd_vel_pub_ = this->create_publisher<geometry_msgs::msg::Twist>("/cmd_v

        current_side_ = 0;
        current_lap_ = 0;
        move_forward_ = true;
        prev_position_ = geometry_msgs::msg::Pose();
        prev_orientation_ = 0.0;
        start_time_ = std::chrono::steady_clock::now();
        last_log_time_ = start_time_;

        RCLCPP_INFO(this->get_logger(),
            "SquareNav initialized with side_length=%.2f, num_laps=%d, clockwise=
            side_length_, num_laps_, clockwise_ ? "true" : "false");
    }

private:
    rclcpp::Subscription<nav_msgs::msg::Odometry>::SharedPtr odom_sub_;
    rclcpp::Subscription<nav_msgs::msg::Odometry>::SharedPtr ground_truth_sub_;
    rclcpp::Publisher<geometry_msgs::msg::Twist>::SharedPtr cmd_vel_pub_;

    double side_length_;
    int num_laps_;
    bool clockwise_;
```

```
    int current_side_;
    int current_lap_;
    bool move_forward_;
    geometry_msgs::msg::Pose prev_position_;
    double prev_orientation_;
    double current_x;
    double current_y;
    double truth_current_x;
    double truth_current_y;
    double current_orientation;
    double truth_current_orientation;
    double current_position_error;
    double current_orientation_error;

    std::vector<double> position_errors_;
    std::vector<double> orientation_errors_;
    std::vector<double> average_position_errors_;
    std::vector<double> average_orientation_errors_;
    std::vector<double> lap_position_errors_;
    std::vector<double> lap_orientation_errors_;
    std::vector<double> times_;


    std::chrono::steady_clock::time_point start_time_;
    std::chrono::steady_clock::time_point last_log_time_;

    void groundTruthCallback(const nav_msgs::msg::Odometry::SharedPtr msg)
    {
        truth_current_x = msg->pose.pose.position.x;
        truth_current_y = msg->pose.pose.position.y;
        truth_current_orientation = getYaw(msg->pose.pose.orientation);

        current_position_error = std::pow(current_x - truth_current_x, 2) + std:
        current_orientation_error = std::pow(current_orientation - truth_current

        position_errors_.push_back(current_position_error);
        orientation_errors_.push_back(current_orientation_error);
        lap_position_errors_.push_back(current_position_error);
        lap_orientation_errors_.push_back(current_orientation_error);

        auto current_time = std::chrono::steady_clock::now();
        std::chrono::duration<double> elapsed_time = current_time - start_time_;

        if (current_time - last_log_time_ >= 10s)
        {
            RCLCPP_INFO(this->get_logger(),
                        "Elapsed time: %f s | Position error: %f | Orientation e
                        elapsed_time.count(), current_position_error, current_or

            times_.push_back(elapsed_time.count());
            last_log_time_ = current_time; // Zaktualizuj czas ostatniego logowa
```

```
        }
    }

    void odomCallback(const nav_msgs::msg::Odometry::SharedPtr msg)
    {
        current_x = msg->pose.pose.position.x;
        current_y = msg->pose.pose.position.y;

        double dx = current_x - prev_position_.position.x;
        double dy = current_y - prev_position_.position.y;
        double distance_travelled = std::sqrt(dx * dx + dy * dy);

        current_orientation = getYaw(msg->pose.pose.orientation);
        double delta_orientation = current_orientation - prev_orientation_;

        if (delta_orientation > M_PI)
        {
            delta_orientation -= 2 * M_PI;
        }
        else if (delta_orientation < -M_PI)
        {
            delta_orientation += 2 * M_PI;
        }

        if (move_forward_)
        {
            if (distance_travelled >= side_length_)
            {
                move_forward_ = false;
                prev_position_ = msg->pose.pose;
                prev_orientation_ = current_orientation;
            }
        }
        else
        {
            double target_rotation = (clockwise_ ? -1 : 1) * M_PI / 2;
            if (std::abs(delta_orientation) >= std::abs(target_rotation))
            {
                move_forward_ = true;
                prev_position_ = msg->pose.pose;
                prev_orientation_ = current_orientation;
                current_side_++;

                if (current_side_ >= 4)
                {
                    current_side_ = 0;
                    current_lap_++;

                    double avg_position_error = std::accumulate(lap_position_err
                    double avg_orientation_error = std::accumulate(lap_orientati
```

```
                    average_position_errors_.push_back(avg_position_error);
                    average_orientation_errors_.push_back(avg_orientation_error)

                    RCLCPP_INFO(this->get_logger(),
                                "Lap %d/%d completed. Avg Position Error: %f | A
                                current_lap_, num_laps_, avg_position_error, avg

                    lap_orientation_errors_.clear();
                    lap_orientation_errors_.clear();
                }
            }
        }

        moveRobot();
        if (current_lap_ >= num_laps_)
        {
            stopRobot();
            return;
        }


    }

    double getYaw(const geometry_msgs::msg::Quaternion &quat)
    {
        double siny_cosp = 2.0 * (quat.w * quat.z + quat.x * quat.y);
        double cosy_cosp = 1.0 - 2.0 * (quat.y * quat.y + quat.z * quat.z);
        return std::atan2(siny_cosp, cosy_cosp);
    }

    void moveRobot()
    {
        auto msg = geometry_msgs::msg::Twist();

        if (move_forward_)
        {
            msg.linear.x = 0.2;
            msg.angular.z = 0.0;
        }
        else
        {
            msg.linear.x = 0.0;
            msg.angular.z = (clockwise_ ? -1 : 1) * 0.3;
        }

        cmd_vel_pub_->publish(msg);
    }

    void stopRobot()
    {
        auto msg = geometry_msgs::msg::Twist();
```

```
        msg.linear.x = 0.0;
        msg.angular.z = 0.0;
        cmd_vel_pub_->publish(msg);
        RCLCPP_INFO(this->get_logger(), "All laps completed. Stopping robot.");
        printSummary();
        rclcpp::shutdown();
    }

    void printSummary()
    {
        std::cout << "\n==================== Temporary Errors ====================\n";
        std::cout << "   Time (s)    |  Position Error   | Orientation Error\n";
        for (size_t i = 0; i < times_.size(); ++i)
        {
            double &time = times_[i];
            double &pos_err = position_errors_[i];
            double &ori_err = orientation_errors_[i];
            std::cout << std::setw(12) << time << " | "
                      << std::setw(16) << pos_err << " | "
                      << std::setw(18) << ori_err << "\n";
        }

        std::cout << "\n=================== Cumulative Errors ====================\n";
        std::cout << "   Lap   | Avg Position Error | Avg Orientation Error\n";
        for (size_t i = 0; i < average_orientation_errors_.size(); ++i)
        {
            double &pos_err = average_position_errors_[i];
            double &ori_err = average_orientation_errors_[i];
            std::cout << std::setw(8) << i + 1 << " | "
                      << std::setw(18) << pos_err << " | "
                      << std::setw(22) << ori_err << "\n";
        }
    }



};

int main(int argc, char *argv[])
{
    rclcpp::init(argc, argv);
    rclcpp::spin(std::make_shared<SquareNav>());
    rclcpp::shutdown();
    return 0;
}
```

Plik CMakeLists

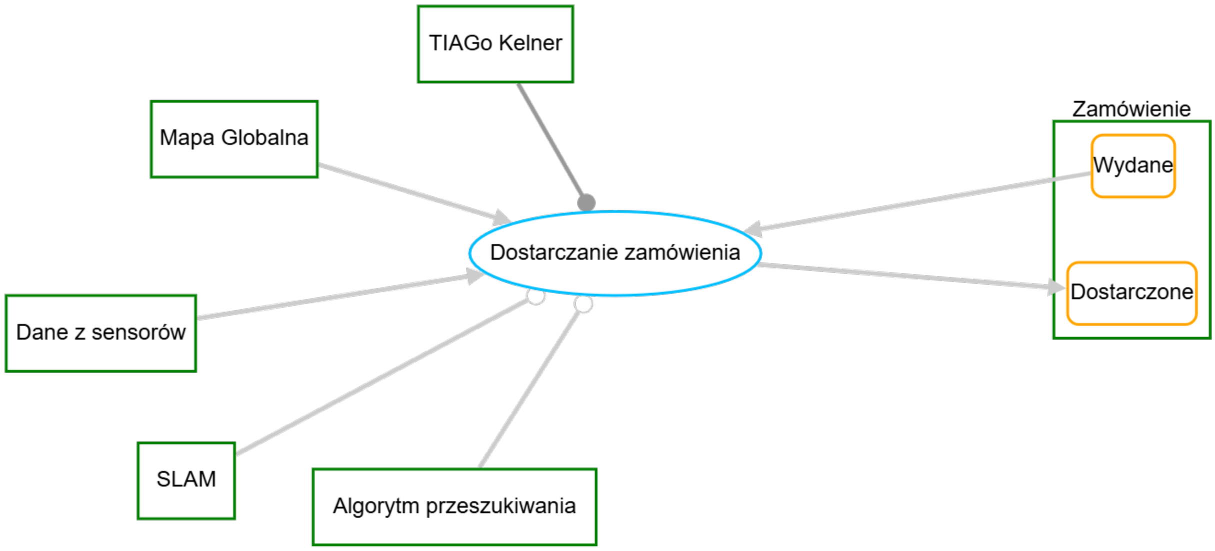
```
cmake_minimum_required(VERSION 3.8)
project(lab4)
```

```
if(CMAKE_COMPILER_IS_GNUCXX OR CMAKE_CXX_COMPILER_ID MATCHES "Clang")
  add_compile_options(-Wall -Wextra -Wpedantic)
endif()

# Znajdź zależności
find_package(ament_cmake REQUIRED)
find_package(rclcpp REQUIRED)
find_package(nav_msgs REQUIRED)
find_package(geometry_msgs REQUIRED)

# Dodaj plik źródłowy square_nav.cpp
add_executable(square_nav src/square_nav.cpp)

# Dodaj katalogi z nagłówkami
target_include_directories(square_nav PUBLIC
  $<BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include>
  $<INSTALL_INTERFACE:include>)

# Wymagania dotyczące standardów C99 i C++17
target_compile_features(square_nav PUBLIC c_std_99 cxx_std_17)

# Połącz target z wymaganymi zależnościami
ament_target_dependencies(square_nav rclcpp nav_msgs geometry_msgs)

# Instalacja pliku wykonywalnego
install(TARGETS square_nav
  DESTINATION lib/${PROJECT_NAME})

# Dodaj plik źródłowy test_nav.cpp
add_executable(test_nav src/test_nav.cpp)

# Dodaj katalogi z nagłówkami
target_include_directories(test_nav PUBLIC
  $<BUILD_INTERFACE:${CMAKE_CURRENT_SOURCE_DIR}/include>
  $<INSTALL_INTERFACE:include>)

# Wymagania dotyczące standardów C99 i C++17
target_compile_features(test_nav PUBLIC c_std_99 cxx_std_17)

# Połącz target z wymaganymi zależnościami
ament_target_dependencies(test_nav rclcpp nav_msgs)

# Instalacja pliku wykonywalnego
install(TARGETS square_nav test_nav
  DESTINATION lib/${PROJECT_NAME})

# Dodaj obsługę testów i linterów
if(BUILD_TESTING)
  find_package(ament_lint_auto REQUIRED)
  # Skoki w linterach (jeśli nie masz jeszcze licencji lub repozytorium git)
  set(ament_cmake_copyright_FOUND TRUE)
```

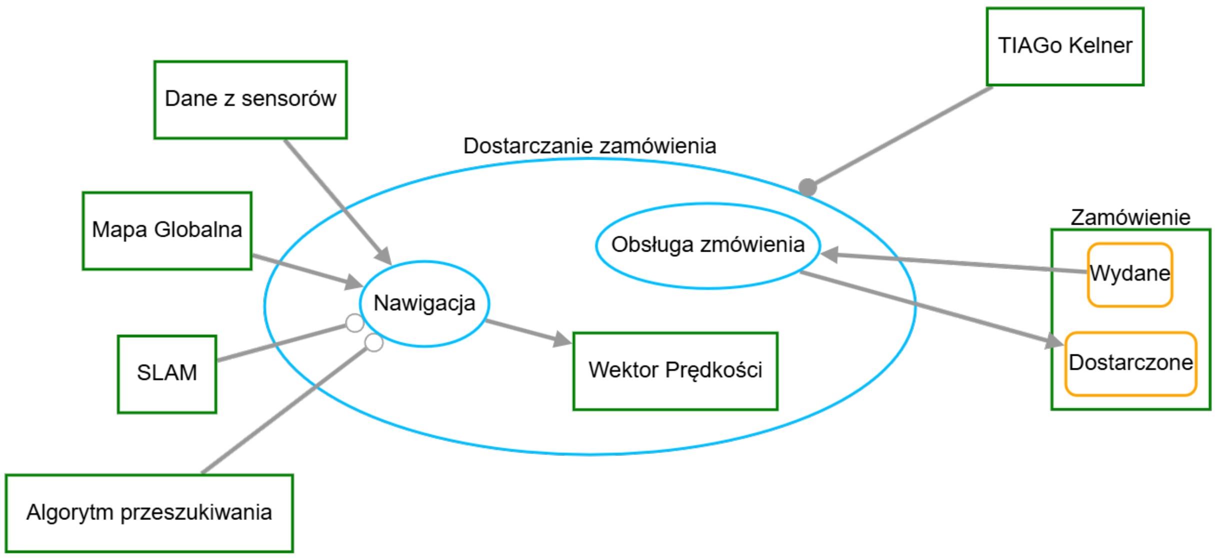
```
  set(ament_cmake_cpplint_FOUND TRUE)
  ament_lint_auto_find_test_dependencies()
endif()
ament_package()
```

2.2. Diagramy OPD robota-kelnera

Diagram poziomu abstrakcji przedstawia przedstawia obiekty potrzebne do zrealizowania procesu dostarczanie zamówienia, a diagram funkcjonalny uszczegóławia proces Dostarczania zamówienia, pokazując wpływ konkretnych obiektów na podprocesy głównego procesu.



Rys. 2.3. Diagram poziomu abstrakcji



Rys. 2.4. Diagram funkcjonalny