

## Отчет по лабораторной работе N4.

Группа:

- 1) Алхименков Леонид
- 2) Казаков Максимилиан
- 3) Хуан Бинкунь



console\_selector\_node.cpp

```
#include <ros/ros.h>
#include <std_msgs/UInt16.h>

void show_menu() {

    std::cout << "\x1B[2J\x1B[H";
    std::cout << "Select algorythm:" << std::endl;
    std::cout << "\t0. Dummy" << std::endl;
    std::cout << "\t1. Voyager" << std::endl;
    std::cout << "\t2. Wall Follower" << std::endl;
    std::cout << "Type -1 to exit" << std::endl;

    std::cout << "Your choose: ";

}

int run_menu() {
    show_menu();
    int choose;
    std::cin >> choose;
    return choose;
}

int main(int argc, char **argv)
{

    ros::init(argc, argv, "console_client");

    ros::NodeHandle node("console_client");

    ros::Publisher algo_pub = node.advertise<std_msgs::UInt16>("/selector", 100);

    for (int choose = 0; choose != -1; choose = run_menu())
    {
        std_msgs::UInt16 msg;
        msg.data = choose;

        algo_pub.publish(msg);

        ros::spinOnce();
    }

    return 0;
}
```

## h control.h

```
#ifndef CONTROL_H
#define CONTROL_H
#include <geometry_msgs/Twist.h>
#include <sensor_msgs/LaserScan.h>

//абстрактный класс - интерфейс системы управления
class Control
{
    //в абстрактном классе обычно нет данных
public:
    //абстрактные функции интерфейса

    //установка данных лазера
    virtual void setLaserData(const std::vector<float>& data) = 0;

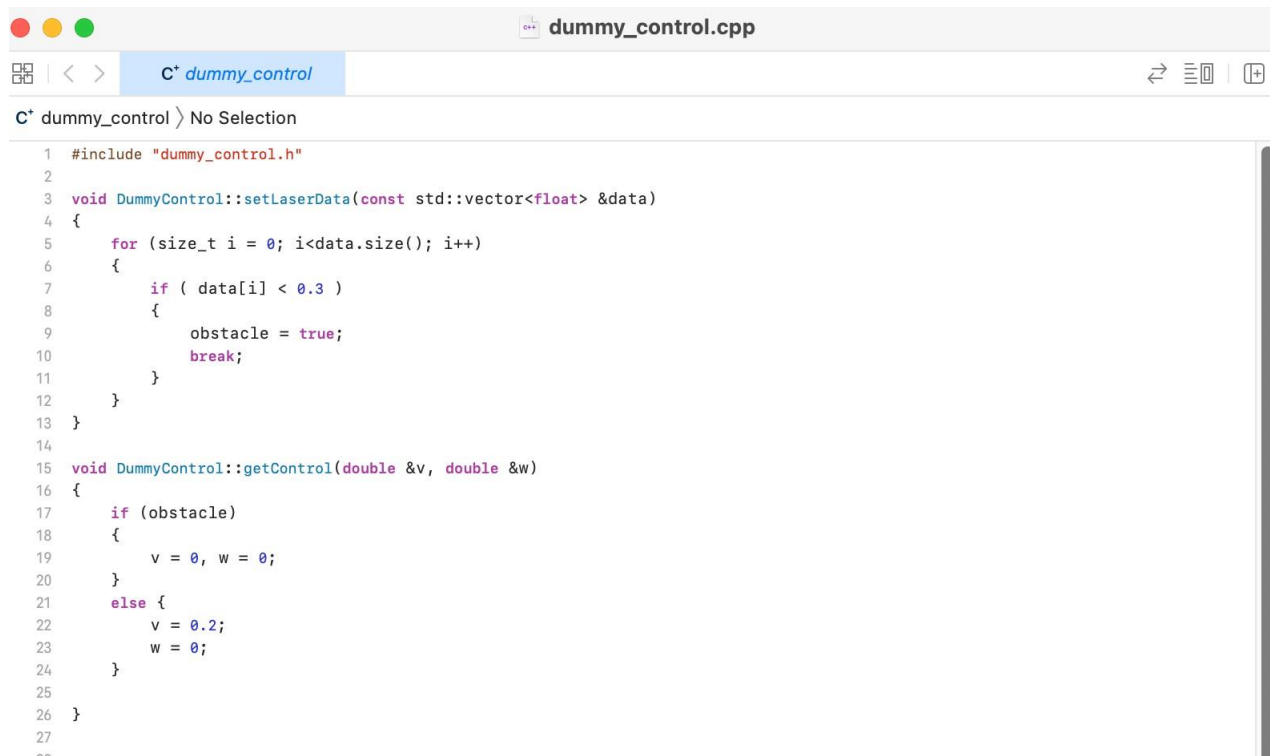
    //установка текущей позиции робота
    virtual void setRobotPose(double x, double y, double theta) = 0;

    //получение управления
    virtual void getControl(double& v, double& w) = 0;

    virtual std::string getName() = 0;

    //виртуальный деструктор
    virtual ~Control() {}
};

#endif // CONTROL_H
```



The screenshot shows a C++ IDE window titled "dummy\_control.cpp". The code implements the Control interface defined in control.h. It includes the necessary headers and implements the setLaserData, setRobotPose, and getControl methods. The setLaserData method checks for obstacles in the laser data and sets a flag. The setRobotPose method sets the robot's position and orientation. The getControl method returns the control values based on the presence of obstacles.

```
1 #include "dummy_control.h"
2
3 void DummyControl::setLaserData(const std::vector<float> &data)
4 {
5     for (size_t i = 0; i<data.size(); i++)
6     {
7         if ( data[i] < 0.3 )
8         {
9             obstacle = true;
10            break;
11        }
12    }
13 }
14
15 void DummyControl::setRobotPose(double x, double y, double theta)
16 {
17     if (obstacle)
18     {
19         v = 0, w = 0;
20     }
21     else {
22         v = 0.2;
23         w = 0;
24     }
25 }
26 }
27
28
```

```
h dummy_control.h

h dummy_control > No Selection

1 #pragma once
2 #include "control.h"
3
4 class DummyControl : public Control
5 {
6 public:
7     //установка данных лазера
8     void setLaserData(const std::vector<float>& data) override;
9
10    //установка текущей позиции робота – для данного вида управления не требуется – поэтому пустая
11    void setRobotPose(double x, double y, double theta) override {}
12
13    //получение управления
14    void getControl(double& v, double& w) override;
15
16    std::string getName() override { return "Dummy"; }
17
18 private:
19     bool obstacle = false;
20 };
21
22
```

```
voyagercontrol.cpp

C* voyagercontrol > No Selection

1 #include "voyagercontrol.h"
2
3
4 void VoyagerControl::setLaserData(const std::vector<float> &data)
5 {
6     obstacle = false;
7     for (size_t i = 0; i<data.size(); i++)
8     {
9         if ( data[i] < min_range )
10        {
11            obstacle = true;
12            ROS_INFO_STREAM("OBSTACLE!!!");
13            break;
14        }
15    }
16 }
17
18 //получение управления
19 void VoyagerControl::getControl(double& v, double& w)
20 {
21     if (obstacle)
22     {
23         v = 0;
24         w = max_omega;
25     }
26     else
27     {
28         v = max_vel;
29         w = 0;
30     }
31 }
32
```

```
h voyagercontrol.h

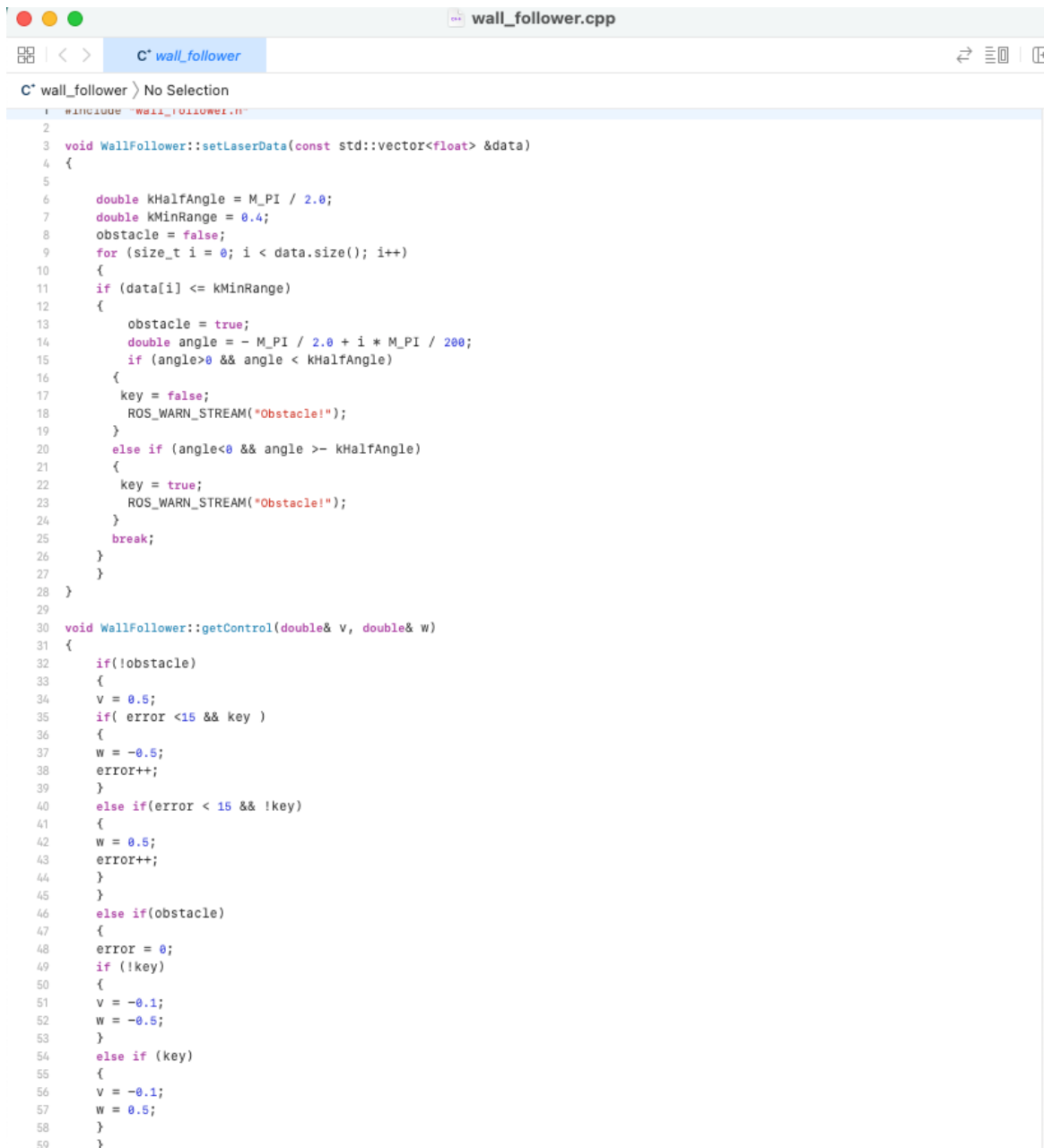
h voyagercontrol > No Selection

1 #pragma once
2
3 #include <ros/ros.h>
4 #include "control.h"
5
6 class VoyagerControl : public Control
7 {
8 private:
9     double min_range;
10    bool obstacle = false;
11    double max_vel;
12    double max_omega;
13
14 public:
15     //установка данных лазера
16     void setLaserData(const std::vector<float>& data) override;
17
18     //установка текущей позиции робота – для данного вида управления не требуется – поэтому пустая
19     void setRobotPose(double x, double y, double theta) override {}
20
21     //получение управления
22     void getControl(double& v, double& w) override;
23
24     std::string getName() override { return "Voyager"; }
25
26     VoyagerControl(double range = 1.0, double maxv = 0.5, double maxw = 0.5):
27         min_range(range),
28         max_vel(maxv),
29         max_omega(maxw)
30     {
31         ROS_DEBUG_STREAM("VoyagerControl constructor");
32     }
33 };
34
```

```
h wall_follower.h

h wall_follower > WallFollower

1 #pragma once
2
3 #include <ros/ros.h>
4 #include "control.h"
5 #include <cmath>
6 class WallFollower : public Control
7 {
8 private:
9     int error = 100 ;
10    double min_range;
11    bool obstacle = false;
12    double max_vel;
13    double min_vel;
14    double max_omega;
15    bool key;
16
17 public:
18     //установка данных лазера
19     void setLaserData(const std::vector<float>& data) override;
20
21     //установка текущей позиции робота – для данного вида управления не требуется – поэтому пустая
22     void setRobotPose(double x, double y, double theta) override {}
23
24     //получение управления
25     void getControl(double& v, double& w) override;
26
27     std::string getName() override { return "NotSoDummy"; }
28     WallFollower(double range = 0.4, double maxv = 0.5, double maxw = 0.5, double minv = 0.1):
29         min_range(range),
30         max_vel(maxv),
31         max_omega(maxw),
32         min_vel(minv)
33     {
34         ROS_DEBUG_STREAM("NotSoDummy constructor");
35     }
36 };
37
38
```



```
1 #include "wall_follower.h"
2
3 void WallFollower::setLaserData(const std::vector<float> &data)
4 {
5
6     double kHalfAngle = M_PI / 2.0;
7     double kMinRange = 0.4;
8     obstacle = false;
9     for (size_t i = 0; i < data.size(); i++)
10     {
11         if (data[i] <= kMinRange)
12         {
13             obstacle = true;
14             double angle = - M_PI / 2.0 + i * M_PI / 200;
15             if (angle>0 && angle < kHalfAngle)
16             {
17                 key = false;
18                 ROS_WARN_STREAM("Obstacle!");
19             }
20             else if (angle<0 && angle >= - kHalfAngle)
21             {
22                 key = true;
23                 ROS_WARN_STREAM("Obstacle!");
24             }
25             break;
26         }
27     }
28 }
29
30 void WallFollower::getControl(double& v, double& w)
31 {
32     if(!obstacle)
33     {
34         v = 0.5;
35         if( error <15 && key )
36         {
37             w = -0.5;
38             error++;
39         }
40         else if(error < 15 && !key)
41         {
42             w = 0.5;
43             error++;
44         }
45     }
46     else if(obstacle)
47     {
48         error = 0;
49         if (!key)
50         {
51             v = -0.1;
52             w = -0.5;
53         }
54         else if (key)
55         {
56             v = -0.1;
57             w = 0.5;
58         }
59     }
```

CMAKELIST

```
cmake_minimum_required(VERSION 2.8.3)
project(control_selector)

add_compile_options(-std=c++11)

find_package(catkin REQUIRED COMPONENTS
  nav_msgs
  roscpp
  std_msgs
)

catkin_package(
  CATKIN_DEPENDS nav_msgs roscpp std_msgs
)

#####
## Build ##
#####
```

```

include_directories(
    ${catkin_INCLUDE_DIRS}
)

## Declare a C++ executable
add_executable(control_selector
    src/control_selector_node.cpp
    src/control.h
    src/dummy_control.h
    src/dummy_control.cpp
    src/voyagercontrol.h
    src/voyagercontrol.cpp
    src/wall_follower.h
    src/wall_follower.cpp
)

## Specify libraries to link a library or executable target against
target_link_libraries(control_selector
    ${catkin_LIBRARIES}
)

## Declare a C++ executable
add_executable(console_selector
    src/console_selector_node.cpp)

## Specify libraries to link a library or executable target against
target_link_libraries(console_selector
    ${catkin_LIBRARIES}
)

```