



**МИНОБРНАУКИ РОССИИ**  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
**«МИРЭА – Российский технологический университет»**  
**РТУ МИРЭА**  
**Институт кибернетики**  
**Кафедра проблем управления**

**Лабораторная работа №4**  
по дисциплине «Программное обеспечение мехатронных и робототехнических систем»

**Тема:** Изучение основ информационного обмена в ROS (Robot operating system)

A handwritten signature in blue ink, appearing to read "Dn Den".

Работу выполнил  
студент КРБО-01-17:  
Денисов Д.С.

Руководитель:  
Морозов А.А.

**Цель работы:** получение навыков установки и настройки ROS (Robot operating system), ознакомление с архитектурой ROS, создание узла, осуществляющего публикацию данных и чтение данных из информационной среды ROS.

**Задание:**

1. Установить и настроить ROS;
2. Создать узел, осуществляющий публикацию данных пользователя;
3. Создать узел-подписчик, принимающий данные пользователя и выводящий их в консоль.

**Ход работы:**

*1. Установка ROS*

Robot operating system - набор инструментов, предназначенных для разработки программных компонентов в области робототехники. В данной работе был установлен фреймворк ROS на персональный компьютер под управлением ОС Windows 10 (версия 10.0.18363, сборка 18363).

В качестве дистрибутива ROS был выбран Noetic Ninjemys. Данная версия является одной из самых актуальных, последней до мажорного изменения ROS2, поддерживает большое количество пакетов. ROS2 на момент выполнения находится в ряде экспериментальных.

В начале, было необходимо установить следующие компоненты:

- среда разработки Visual Studio 2019, которая используется для компиляции пакетов ROS;
- менеджер пакетов chocolatey, который для операционной системы Windows является заменой инструмент apt-get.

Для установки фреймворка в командной строке были введены инструкции, приведённые на рисунке 1. После выполнения команды, на компьютере были установлены ROS, наиболее популярные библиотеки, среда моделирования робототехнических систем Gazebo, rqt, rviz, а также Python 3.8.

```
mkdir c:\opt\chocolatey
set ChocolateyInstall=c:\opt\chocolatey
choco source add -n=ros-win -s="https://aka.ms/ros/public" --priority=1
choco upgrade ros-noetic-desktop_full -y --execution-timeout=0
```

Рис. 1 - Инструкции для установки ROS

## 2. Настройка ROS

Далее были проведены шаги по настройке ROS, упрощающие с ним работу.

Во-первых, был создан ярлык, который запускает командную строку с загрузкой всех требуемых переменных окружения (рисунок 2).

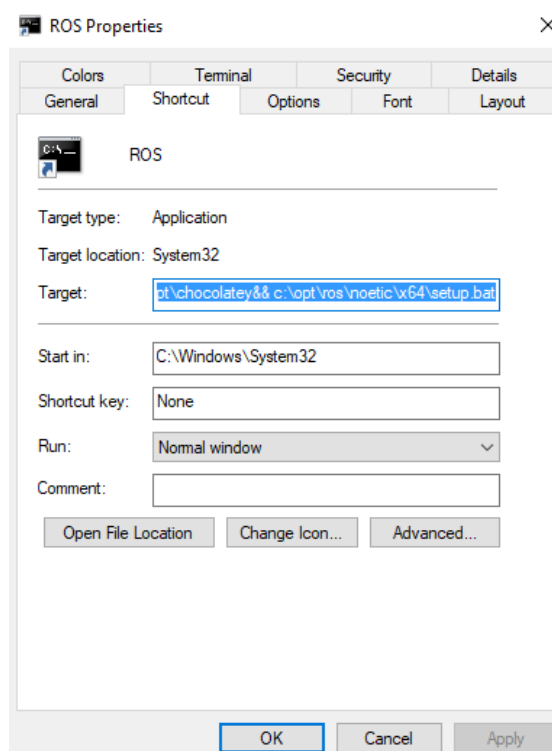


Рис. 2 - Путь ярлыка для запуска ROS

Во-вторых, было решено настроить среду так, чтобы переменные среды ROS автоматически добавлялись в сеанс PowerShell каждый раз при запуске нового терминала. Это является полезным при использовании Visual Studio Code в качестве среды разработки, - она позволяет создавать интегрированные PowerShell терминалы для запуска нескольких узлов.

В Windows Powershell имеются профили – наборы инструкций, выполняющиеся при каждом запуске терминала. В качестве маркера для

того, чтобы определить, нужно ли загружать переменные, используется файл рабочей директории «.catkin\_workspace». Он присутствует в корне каждого ROS-проекта. Проверка по маркеру предотвращает от выполнение длинного списка инициализирующих скриптов, когда это не требуется. Скрипт приведён на рисунке 3.

```
if (Test-Path ".catkin_workspace") {  
    & "C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\Tools\VsDevCmd.bat" -arch=amd64 -host_arch=amd64  
    set ChocolateyInstall="c:\opt\chocolatey"  
    "c:\opt\ros\noetic\x64\setup.bat"  
    clear  
    echo "==== ROS powered ==="  
}
```

Рис. 3 - Скрипт для загрузки переменных окружения ROS при запуске терминала

### *3. Реализация информационного обмена*

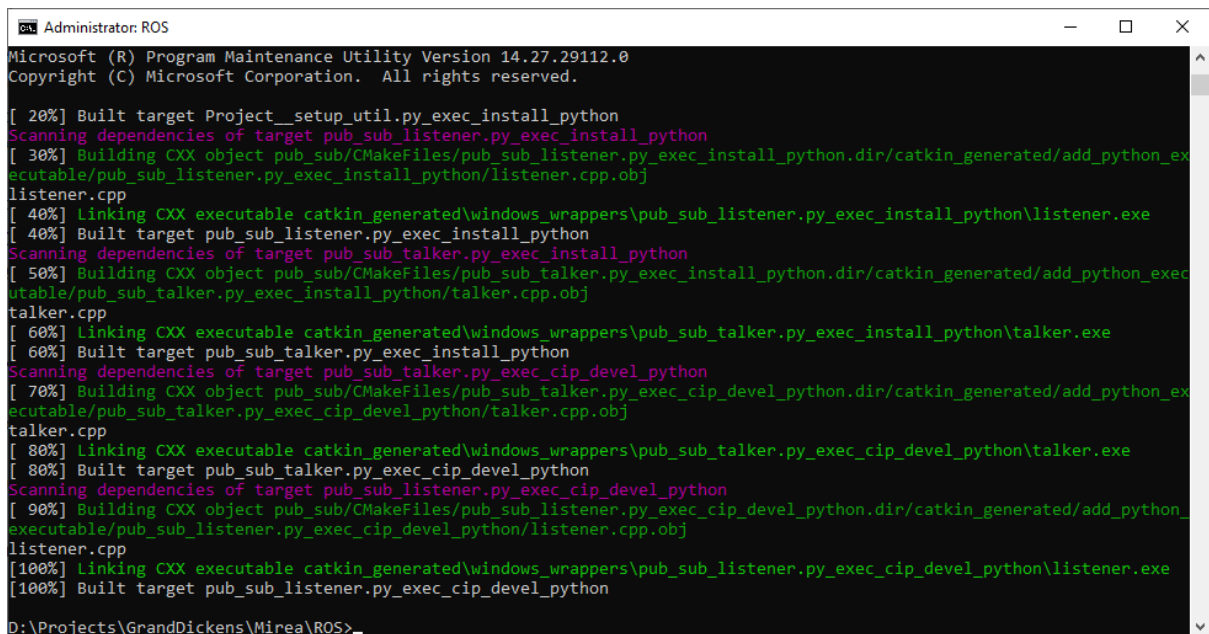
С помощью команды «catkin\_make» была создана новая рабочая директория, а в ней – пакет с названием «pub\_sub».

Затем был создан новый тип сообщения, который хранит информацию о пользователе: имя, фамилию и отчество, и выставлены все необходимые зависимости в файлах «package.xml» и «CMakeLists.txt».

Был создан новый файл «talker.py», внутри которого на языке программирования Python был реализован алгоритм по публикации данных об определённом пользователе. Публикация осуществляется в топик «current\_user» с частотой 1 Гц. Листинг программы приведён в приложении А.

После этого в файле «listener.py» был создан второй узел. Его задачей является получение данных из топика «current\_user» и отображение информации в виде фамилии и инициалов пользователя с добавлением системного времени. С листингом программы можно ознакомиться в приложении Б.

Командой «catkin\_make» были сгенерированы файлы, описывающие тип сообщения, а также проведена индексация реализованных узлов (рисунок 4).



```
Administrator: ROS
Microsoft (R) Program Maintenance Utility Version 14.27.29112.0
Copyright (C) Microsoft Corporation. All rights reserved.

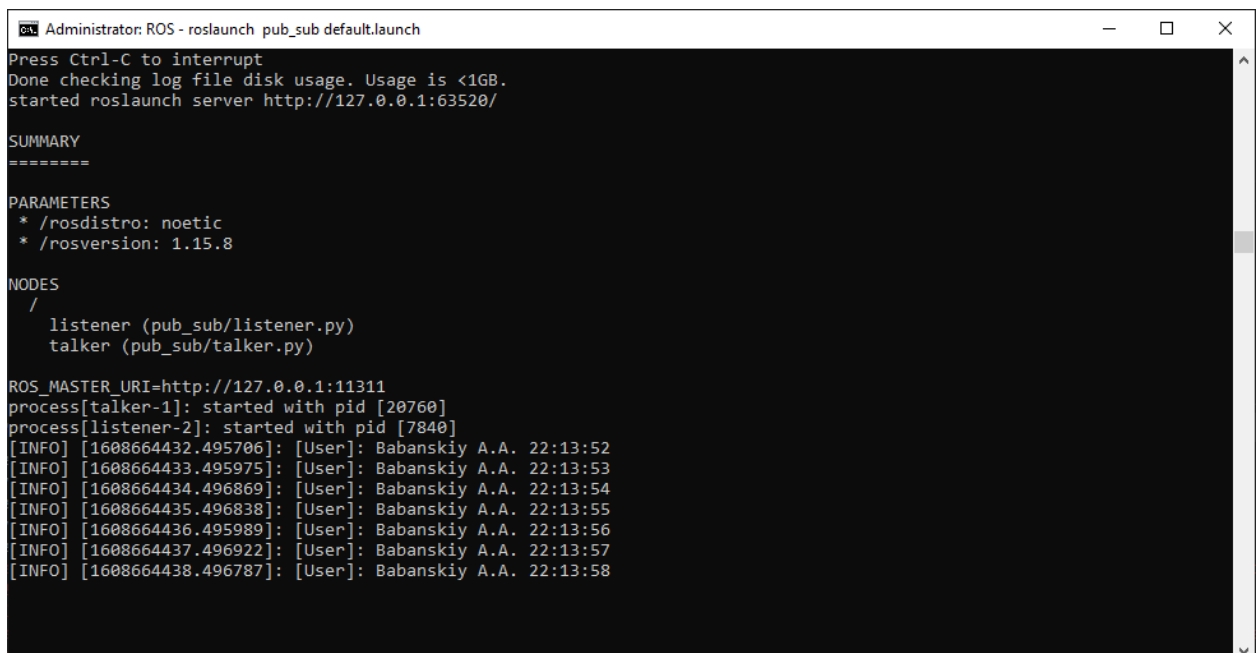
[ 20%] Built target Project__setup_util.py_exec_install_python
Scanning dependencies of target pub_sub_listener.py_exec_install_python
[ 30%] Building CXX object pub_sub/CMakeFiles/pub_sub_listener.py_exec_install_python.dir/catkin_generated/add_python_executable/pub_sub_listener.py_exec_install_python/listener.cpp.obj
listener.cpp
[ 40%] Linking CXX executable catkin_generated/windows_wrappers/pub_sub_listener.py_exec_install_python\listener.exe
[ 40%] Built target pub_sub_listener.py_exec_install_python
Scanning dependencies of target pub_sub_talker.py_exec_install_python
[ 50%] Building CXX object pub_sub/CMakeFiles/pub_sub_talker.py_exec_install_python.dir/catkin_generated/add_python_executable/pub_sub_talker.py_exec_install_python/talker.cpp.obj
talker.cpp
[ 60%] Linking CXX executable catkin_generated/windows_wrappers/pub_sub_talker.py_exec_install_python\talker.exe
[ 60%] Built target pub_sub_talker.py_exec_install_python
Scanning dependencies of target pub_sub_talker.py_exec_cip_devel_python
[ 70%] Building CXX object pub_sub/CMakeFiles/pub_sub_talker.py_exec_cip_devel_python.dir/catkin_generated/add_python_executable/pub_sub_talker.py_exec_cip_devel_python/talker.cpp.obj
talker.cpp
[ 80%] Linking CXX executable catkin_generated/windows_wrappers/pub_sub_talker.py_exec_cip_devel_python\talker.exe
[ 80%] Built target pub_sub_talker.py_exec_cip_devel_python
Scanning dependencies of target pub_sub_listener.py_exec_cip_devel_python
[ 90%] Building CXX object pub_sub/CMakeFiles/pub_sub_listener.py_exec_cip_devel_python.dir/catkin_generated/add_python_executable/pub_sub_listener.py_exec_cip_devel_python/listener.cpp.obj
listener.cpp
[100%] Linking CXX executable catkin_generated/windows_wrappers/pub_sub_listener.py_exec_cip_devel_python\listener.exe
[100%] Built target pub_sub_listener.py_exec_cip_devel_python

D:\Projects\GrandDickens\Mirea\ROS>
```

Рис. 4 – Результаты компиляции пакета

Для упрощения работы также был создан launch-файл, запускающий оба узла.

В результате выполнения команды «`roslaunch pub_sub default.launch`» был запущен узел-издатель и узел-подписчик, который выводит в консоль информацию, поступающую каждую секунду (рисунок 5).



```
Administrator: ROS - roslaunch pub_sub default.launch
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://127.0.0.1:63520/

SUMMARY
=====

PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.15.8

NODES
/
  listener (pub_sub/listener.py)
  talker (pub_sub/talker.py)

ROS_MASTER_URI=http://127.0.0.1:11311
process[talker-1]: started with pid [20760]
process[listener-2]: started with pid [7840]
[INFO] [1608664432.495706]: [User]: Babanskiy A.A. 22:13:52
[INFO] [1608664433.495975]: [User]: Babanskiy A.A. 22:13:53
[INFO] [1608664434.496869]: [User]: Babanskiy A.A. 22:13:54
[INFO] [1608664435.496838]: [User]: Babanskiy A.A. 22:13:55
[INFO] [1608664436.495989]: [User]: Babanskiy A.A. 22:13:56
[INFO] [1608664437.496922]: [User]: Babanskiy A.A. 22:13:57
[INFO] [1608664438.496787]: [User]: Babanskiy A.A. 22:13:58
```

Рис. 5 - Вывод опубликованных сообщений в консоль

**Вывод:** были получены навыки установки и настройки ROS (Robot operating system), реализован информационный обмен между узлами на основе модели издатель/подписчик.

## Приложение А

### Листинг программы узла-издателя

```
#!/usr/bin/env python
import rospy
import roslib
from pub_sub.msg import User

def talker():
    pub = rospy.Publisher('current_user', User, queue_size=10)
    rospy.init_node('user_publisher', anonymous=True)
    rate = rospy.Rate(1)
    while not rospy.is_shutdown():
        user = User()
        user.first_name = "Alexey"
        user.middle_name = "Antonovich"
        user.last_name = "Babanskiy"
        pub.publish(user)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

## Приложение Б

### Листинг программы узла-подписчика

```
#!/usr/bin/env python
import rospy
import roslib
from datetime import datetime
from pub_sub.msg import User

def callback(user):
    user_str = "{ } { }.{ }.".format(
        user.last_name, user.first_name[0], user.middle_name[0])
    #rospy.loginfo(rospy.get_caller_id() + "I heard %s", data.data)
    rospy.loginfo("[User]: {user} {time:%H:%M:%S}".format(
        user=user_str, time=datetime.now()))

def listener():
    rospy.init_node('user_listener', anonymous=True)
    rospy.Subscriber("current_user", User, callback)
    rospy.spin()

if __name__ == '__main__':
    listener()
```