Linux

http://learn.voltbro.ru/free/ros-intro/lesson1/about-linux.html

Командная строка и основные команды Linux

Открытие терминала Ctrl + Alt + T

Tab - автодополнение;

- **Стрелка вверх** - предыдущая команда в истории;
- **Стрелка вниз** - следующая команда в истории;
- **Ctrl + Alt + C** - скопировать выделенный фрагмент текста
- **Ctrl + Alt + V** - скопировать в терминал из буфера обмена
- **Ctrl + C** - остановить выполняющуюся программу
- **Ctrl + D** - завершить текущий сеанс связи
- Ctrl + Shift + C `скопировать выделенный текст`
- Ctrl + Shift + V `вставить`

Специальные символы

- `.` текущая директория
- `..` директория на уровень выше
- ~ домашняя директория
- `*` любое количество любых символов
- `?` ровно один любой символ

`Команды терминала Linux`

1. ls вывода списка файлов

2. cd - сменить директорию)

3. mkdir - создать директорию

4. touch - можно создать пустой файл.

5. mv - переместить

6. cp – скопировать

7. rm - удалить

8. sudo (Substitute User & DO - подменить пользователя и выполнить

9. find (Найти) предназначена для поиска файлов.

9. cat - связывать

10. nano консольный текстовый редактор.

11. | `ls /usr/bin | more`

12. ssh утилита подключения к удаленному компьютеру.

13. scp позволяет копировать файлы с одного компьютера на другой по сети.

```
scp  user@ubuntu-pc1:/home/rosuser/robot.dat pi@turtlebro35.local:/home/pi/robot.dat
```

14. grep сортирует и фильтрует текст

Синтаксис: grep [опции] шаблон [имя файла...] или: команда | grep [опции] шаблон

15. apt `или apt-get менеджер пакетов`, который служит для установки и удаления программ (пакетов), обновления системы.

Например, чтобы установить консольный файловый менеджер Midnight Commander, выполните:

```
sudo apt install mc
```

Если apt недоступен, то используйте apt-get:

```
sudo apt-get install mc
```

Команда chmod - изменение прав доступа

chmod a+x name-File

`test1` http://learn.voltbro.ru/free/ros-intro/lesson2/about-python.html

`// Tutorial //`

1 Установка Python 3 и настройка среды программирования на сервере Ubuntu 20.04

https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-programming-environment-on-an-ubuntu-20-04-server-ru

Сочетание	Что делает nano
Ctrl+A	Переместить курсор в начало строки.
Ctrl+E	Переместить курсор в конец строки.
Ctrl+Y	Переместить курсор на 1 страницу вверх (аналог PageUp)
Ctrl+V	Переместить курсор на 1 страницу вниз (аналог PageDown)
Ctrl+_	Перейти к определенной строке (после нажатия введите номер строки).
Ctrl+C	Показать на какой строке и в какой позиции находится курсор.
Ctrl+W	Поиск текста в файле (после нажатия введите что искать).
Ctrl+\	Поиск и замена текста в файле (после нажатия введите что найти/заменить, а после на что менять).
Ctrl+D	Удалить символ под курсором.
Ctrl+K	Удалить текущую строку.
Ctrl+O	Сохранить изменения, не закрывая редактор.
Ctrl+X	Выход из редактора. Если файл был изменен, появится запрос на сохранение изменений.

Работа в python3

В своей директории (Kolodkin) создали директорию test-py.

cd test-py nano test1.py python3 test1.py

Программа в ROS

Основы

Сообщения -

Типовые сообщения ROS, находятся в пакете `std_msgs`. Мы можем посмотреть все типы сообщений этого пакета.

```
rosmsg package std_msgs
```

```
Все сообщения доступные в системе, мы можем посмотреть командой
```

```
rosmsg list
```

Посмотреть информацию о структуре сообщения возможно используя параметр `info` и имя типа сообщения. Для данных датчика `IMU` (инерциальный датчик) мы получим структуру сообщения как представлено ниже.

```
rosmsg info sensor_msgs/Imu
```

Консольная утилита rostopic

rostopic это специальная консольная утилита, предназначенная для отображения отладочной информации о топиках в ROS. С ее помощью удобно искать нужные топики, и выводить сообщения в консоль для отладки.

Список основных используемых команд:

```
rostopic bw     Показать занимаемый сетевой канал
rostopic echo   Вывести сообщения на экран
rostopic find   Поиск топика по типу
rostopic hz     Показать частоту обновления топика
rostopic info   Показать информацию о топике
rostopic list   Показать список существующий топиков
rostopic pub    Опубликовать данные в топик
rostopic type   Показать тип сообщения для топика
```

Консольная утилита rosservice

Для отладки и тестирования сервисов ROS существует специальная консольная утилита rosservice:

```
rosservice call Выполнение запроса к серверу
rosservice find Поиск сервиса по типу
rosservice info Выводит информацию о сервисе
rosservice list Выводит список запущенных сервисов
rosservice type Выводит тип сообщений используемый сервисом
rosservice uri  Выводит RPC URL сервиса
```

Плагин SSH FS редактора VSCode.

Запись на роботе -19 (с терминала Ctrl+Alt+T)

ssh pi@turtlebro19.local

Password brobro

ls

Сохраним lesson на роботе (именно на нем мы хотим считать температуру) в файл/home/pi/base-ros/lesson2/temp_topic_publisher

cd base-ros/

mkdir lesson2/

cd lesson2

python3 lesson1-230221

```
#! /usr/bin/env python3

# -*- coding: utf-8 -*-
```

```
import rospy
from std_msgs.msg import Float32

rospy.init_node('temp_publisher')
pub = rospy.Publisher('/cpu_temp', Float32, queue_size=1)
rate = rospy.Rate(1)
temp = Float32()


def getCPUTemp():
    data = open('/sys/class/thermal/thermal_zone0/temp', 'r').read()
    return round(float(int(data)/1000.0),1)

while not rospy.is_shutdown():
    temp.data = getCPUTemp()
    pub.publish(temp)
    rate.sleep()
```

После запуска программы в терминале, запускаем второй терминал, переходим на робот ssh pi@turtlebro19.local с паролем brobro

pi@turtlebro19:~ $ rostopic list Смотрим список топиков, Должен быть топик /cpu_temp Подаем команду печати содержимого топика

pi@turtlebro19:~ $ rostopic echo /cpu_temp В терминале печатаются значения температуры.

Работа по сети

В рамках одной системы, должна существовать только одна нода `roscore`. Адрес этой ноды `localhost`

Для компьютера, на котором не запущен `roscore`, мы должны установить значение переменной окружения `ROS_MASTER_URI` используя команду `export`

```
export ROS_MASTER_URI=http://192.168.0.145:11311
```

Где 192.168.0.145 - это IP-адрес вашего робота.

Еще один экспорт необходимо сделать для того, чтобы робот мог отсылать данные на ваш ПК.

```
export ROS_HOSTNAME=192.168.0.111
```

Где 192.168.0.111 - это IP-адрес вашего ПК.

Обе команды `export` необходимо выполнить в терминале на вашем ПК, указав "ваши" IP-адреса.

Если все правильно, то при выполнении любой команды ROS на вашем локальном компьютере вы будете обращаться к RaspberryPI робота. Например, команда `rostopic list` выведет список топиков робота.

Управление роботом

ssh pi@turtlebro19.local

password brobro

rostopic info cmd_vel rostopic info cmd_vel

rosmsg show geometry_msgs/Twist

rostopic pub /cmd_vel geometry_msgs/Twist

Уперавление Черепахой

https://robocraft.ru/technology/509

1) Запускаем в терминале roscore

2) Запускаем во втором терминале rosrun turtlesim turtlesim_node

3) Запускаем ~/Kolodkin/test_package/turless(230329).py

Программы (Python)

Движение черепахи по квадрату (turtles-230329.py)

```
import rospy
import math
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
rospy.init_node("mover")
first_run = True
current_pose = Pose()
init_pose = Pose()
side_len = 1
pub = rospy.Publisher("/turtle1/cmd_vel", Twist, queue_size=10)
def odom_cb(pose):
    global current_pose
    current_pose = pose
rospy.Subscriber("/turtle1/pose", Pose, odom_cb)
def forward(des_dist):
    global init_pose, current_pose, first_run, side_len
    init_pose = current_pose
    dist = math.sqrt((init_pose.x-current_pose.x)**2 + (init_pose.y-current_pose.y)**2)
    while dist < des_dist:
        dist = math.sqrt((init_pose.x-current_pose.x)**2 + (init_pose.y-current_pose.y)**2)
        publish_vel(1,0)
    else:
        publish_vel(0,0)
def turn(des_angle,agl):
    global init_pose, current_pose, first_run, side_len
    angle_turn = True
    while angle_turn:
        if abs(init_pose.theta - current_pose.theta) < des_angle:
            if agl > 0:
                publish_vel(0,0.2)
            else:
                publish_vel(0,-0.2)
```

```
            else:
                angle_turn = False
                publish_vel(0,0)
def publish_vel(vel_x, vel_z):
    vel = Twist()
    vel.linear.x = vel_x
    vel.angular.z = vel_z
    pub.publish(vel)
rospy.sleep(0.2)
for i in range(4):
    forward(1)
    turn(math.pi/2,math.pi/2)
```

Движение черепахи по восьмиугольнику

```
import rospy
import math
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose
rospy.init_node("mover")
first_run = True
current_pose = Pose()
init_pose = Pose()
side_len = 1
nn = 8
pub = rospy.Publisher("/turtle1/cmd_vel", Twist, queue_size=10)
def odom_cb(pose):
    global current_pose
    current_pose = pose
rospy.Subscriber("/turtle1/pose", Pose, odom_cb)
def forward(des_dist):
    global init_pose, current_pose, first_run, side_len
    init_pose = current_pose
    dist = math.sqrt((init_pose.x-current_pose.x)**2 + (init_pose.y-current_pose.y)**2)
    while dist < des_dist:
        dist = math.sqrt((init_pose.x-current_pose.x)**2 + (init_pose.y-current_pose.y)**2)
        publish_vel(1,0)
    else:
        publish_vel(0,0)
def turn(des_angle,agl):
    global init_pose, current_pose, first_run, side_len
    angle_turn = True
    while angle_turn:
        if abs(init_pose.theta - current_pose.theta) < des_angle:
            if agl > 0:
                publish_vel(0,0.1)
            else:
                publish_vel(0,-0.1)
        else:
            angle_turn = False
```

```
                publish_vel(0,0)
def publish_vel(vel_x, vel_z):
    vel = Twist()
    vel.linear.x = vel_x
    vel.angular.z = vel_z
    pub.publish(vel)
rospy.sleep(0.2)
dd = 2.0*math.pi/nn
for i in range(nn):
    forward(1)
    turn(dd,dd)
```

Движение робота

Подключаем робота ssh pi@turtlebro19.local

Переносим программу с компьютера на робот и запускаем ее на роботе

№ 1. Программа движения робота по квадрату -

```
import rospy
import math
from geometry_msgs.msg import Twist
from geometry_msgs.msg import Pose2D
rospy.init_node("mover")

first_run = True
current_Pose2D = Pose2D()
init_Pose2D = Pose2D()
nn = 4
side_len = 1/nn
pub = rospy.Publisher("/cmd_vel", Twist, queue_size=10)
def odom_cb(Pose2D):
    global current_Pose2D
    current_Pose2D = Pose2D
rospy.Subscriber("/odom_pose2d", Pose2D, odom_cb)
def forward(des_dist):
    global init_Pose2D, current_Pose2D, first_run, side_len
    init_Pose2D = current_Pose2D
    dist = math.sqrt((init_Pose2D.x-current_Pose2D.x)**2 + (init_Pose2D.y-current_Pose2D.y)**2)
    while dist < des_dist:
        dist = math.sqrt((init_Pose2D.x-current_Pose2D.x)**2 + (init_Pose2D.y-current_Pose2D.y)**2)
        publish_vel(0.25,0)
    else:
```

```
            publish_vel(0,0)
def turn(des_angle,agl):
        global init_Pose2D, current_Pose2D, first_run, side_len
        angle_turn = True
        while angle_turn:
            if abs(init_Pose2D.theta - current_Pose2D.theta) < des_angle:
                if agl > 0:
                    publish_vel(0,0.1)
                else:
                    publish_vel(0,-0.1)
            else:
                angle_turn = False
                publish_vel(0,0)
def publish_vel(vel_x, vel_z):
        vel = Twist()
        vel.linear.x = vel_x
        vel.angular.z = vel_z
        pub.publish(vel)
rospy.sleep(0.2)
dd = 2.0*math.pi/nn
l = side_len
for i in range(nn):
        forward(l)
        turn(dd,dd)
```

№ 2. Программа движения робота по змейке (move_shake_230510.py)

```
# Движение по змейке (ломанная линия. Шаг движения side_len = 1/nn)

import rospy
import math
from geometry_msgs.msg import Twist
from geometry_msgs.msg import Pose2D
rospy.init_node("mover")
first_run = True
current_Pose2D = Pose2D()
init_Pose2D = Pose2D()
nn = 4
side_len = 1/nn
pub = rospy.Publisher("/cmd_vel", Twist, queue_size=10)
def odom_cb(Pose2D):
            global current_Pose2D
            current_Pose2D = Pose2D
rospy.Subscriber("/odom_pose2d", Pose2D, odom_cb)
def forward(des_dist):
            global init_Pose2D, current_Pose2D, first_run, side_len
            init_Pose2D = current_Pose2D
            dist = math.sqrt((init_Pose2D.x-current_Pose2D.x)**2 + (init_Pose2D.y-current_Pose2D.y)**2)
            while dist < des_dist:
```

```
                dist = math.sqrt((init_Pose2D.x-current_Pose2D.x)**2 + (init_Pose2D.y-
                                                                current_Pose2D.y)**2)
                publish_vel(0.25,0)
        else:
                publish_vel(0,0)
def turn(des_angle,agl):
        global init_Pose2D, current_Pose2D, first_run, side_len
        angle_turn = True
        while angle_turn:
                if abs(init_Pose2D.theta - current_Pose2D.theta) < des_angle:
                        if agl > 0:
                                publish_vel(0,0.1)
                        else:
                                publish_vel(0,-0.1)
                else:
                        angle_turn = False
                        publish_vel(0,0)
def publish_vel(vel_x, vel_z):
        vel = Twist()
        vel.linear.x = vel_x
        vel.angular.z = vel_z
        pub.publish(vel)
rospy.sleep(0.2)
dd = 2.0*math.pi/nn
l = side_len
nn1 = int(nn/2)
for i in range(nn1):
        forward(l)
        turn(dd,dd)
        forward(l)
        turn(dd,-dd)
```

№ 3. Программа движения робота до стенки -

```
import rospy

from sensor_msgs.msg import LaserScan

from geometry_msgs.msg import Twist
rospy.init_node('dalnomer')
target = 0.5
vel = Twist()
pub = rospy.Publisher("cmd_vel", Twist, queue_size = 1)
def callback(scan):
        if scan.ranges[0] > target:
                vel.linear.x = 0.1
        else:
                vel.linear.x = 0
pub.publish(vel)
rospy.Subscriber("/scan", LaserScan, callback)
rospy.spin()
```

№ 3-а. Программа дальномер - 1

```
import rospy
import math
# Модуль для работы с сообщением из топика /scan (топик лидара)
from sensor_msgs.msg import LaserScan
rospy.init_node('dalnomer')
def callback(scan):
 #   Печать расстояния до преграды
        print(scan.ranges[0])
rospy.Subscriber("/scan", LaserScan, callback)
rospy.spin()
```

№ 3-b. Программа дальномер - 2

```
import rospy
# Модуль для работы с сообщением из топика /scan (топик лидара)
from sensor_msgs.msg import LaserScan
# Модуль для работы с сообщениями топика /cmd_vel (движение робота)
from geometry_msgs.msg import Twist
import math

import rospy
import math
from sensor_msgs.msg import LaserScan
rospy.init_node('dalnomer')
def callback(scan):
print(scan.ranges[0])
rospy.Subscriber("/scan", LaserScan, callback)
rospy.spin()
# Подключение к топику управления скоростью движения робота
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)

def callback(scan: LaserScan):
        # Индекс угла в массиве ranges
        idx = int(math.radians(90)/scan.angle_increment)
        # расстояние до объектов по оси X
        # впереди робота
        distance_90 = scan.ranges[idx]
        distance_0 = scan.ranges[0]
        print(f'Distance_90: {distance_90}, distance_0: {distance_0}')
        vel = Twist()
        # Создаем условие дивжения
        # Если расстояние до препятствия больше 20 см, двигаться
        # Иначе остановиться
        if (distance_90 <= 0.5 and distance_90 != float('inf')) and distance_0 > 0.5:
        # if distance_0 > 0.3:
        # Задаем линейную скорость движения
                vel.linear.x = 1
        else:
```

```
                vel.linear.x = 0

        # Отправляем команду о движении роботу
        pub.publish(vel)
rospy.Subscriber("/scan", LaserScan, callback)
rospy.spin()
```

4. Программа движения робота на 1 метр вперед

```
import rospy

from nav_msgs.msg import Odometry
from geometry_msgs.msg import Twist
rospy.init_node('lineyka')
target = 1        # move on 1 m
vel = Twist()
pub = rospy.Publisher("cmd_vel", Twist, queue_size = 1)
def callback(odom):
        if odom.pose.pose.position.x < target:
                vel.linear.x = 0.1
        else:
                vel.linear.x = 0
        pub.publish(vel)
rospy.Subscriber("/odom", Odometry, callback)
rospy.spin()
```

5. Программа движения робота по линии

```
import rospy, math
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
rospy.init_node('line_mover')
pub = rospy.Publisher("/cmd_vel", Twist, queue_size=1)
pub_vel = Twist()
pub_vel.linear.x = 0.05 #будем всегда ехать вперед
def quaternion_to_theta(odom):
        return(2*(math.asin(odom.pose.pose.orientation.z)*
                        math.copysign(1,odom.pose.pose.orientation.w)))
def odomcb(odom):
        theta = quaternion_to_theta(odom)
        pub_vel.angular.z = - theta
        pub.publish(pub_vel)
rospy.Subscriber("/odom", Odometry, odomcb)
```

6. Программа робот транспортир

```
# Робот транспортир 230517
import rospy
import math
```

```
from nav_msgs.msg import Odometry
rospy.init_node('transportir')
def callback(odom):
        print(2*math.degrees(math.asin(odom.pose.pose.orientation.z)*
                math.copysign(1,odom.pose.pose.orientation.w)))
rospy.Subscriber("/odom", Odometry, callback)
rospy.spin()
```

7. Программа Управление роботом - По прямой любой ценой

```
import rospy, math
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
rospy.init_node('line_mover')
pub = rospy.Publisher("/cmd_vel", Twist, queue_size=1)
pub_vel = Twist()
pub_vel.linear.x = 0.05   #будем всегда ехать вперед
def quaternion_to_theta(odom):
        return(2*(math.asin(odom.pose.pose.orientation.z)*
                        math.copysign(1,odom.pose.pose.orientation.w)))
def odomcb(odom):
        theta = quaternion_to_theta(odom)
        pub_vel.angular.z = - theta
        pub.publish(pub_vel)
rospy.Subscriber("/odom", Odometry, odomcb)
rospy.spin()
```