Инженерный вестник Дона, №1 (2024)
ivdon.ru/ru/magazine/archive/n1y2024/8934

Взаимодействие Tello EDU и Turtlebot3 в помещениях со слабым сигналом при помощи локальной сети и операционной системы ROS

И.А. Хорсик[1], О.С. Амосов[2], С.Г. Амосова[2], Л.В. Бунина[1].

[1]МИРЭА — Российский технологический университет

[2]Институт проблем управления им. В. А. Трапезникова Российской академии наук (ИПУ РАН)

Аннотация: В статье исследуется взаимодействие Tello EDU, малогабаритного образовательного дрона, с Turtlebot3, беспилотным наземным аппаратом, в помещениях со слабым сигналом. Показано, как с использованием локальной сети и операционной системы роботов (ROS) можно достичь эффективной совместной работы этих двух устройств. Анализируется, как локальная сеть может быть использована для трансляции данных и контроля устройств в условиях слабого внешнего сигнала. Исследуется роль ROS как основного инструмента для управления и взаимодействия с устройствами. Кроме того, статья рассматривает конкретные сценарии использования, включая взаимодействие и координацию между Tello EDU и Turtlebot3. Также представлена схема взаимодействия двух беспилотных аппаратов, описано подробное описание их работы, и представлен код на Python с применением различных библиотек на основе роботизированной операционной системы ROS.

Ключевые слова: Tello Edu, операционная система (ROS), БПЛА, локальная сеть, Wi-fi, узлы (nodes), SLAM, слабый сигнал, планирование маршрута, автономный робот, Turtlebot3.

Введение

В мире робототехники постоянно исследуются и разрабатываются новые подходы к управлению и координации роботов в различных условиях. Одним из важных аспектов исследования является работа роботов в помещениях со слабым сигналом, где традиционные методы беспроводной связи могут быть неэффективны.

Основным фокусом настоящего научного исследования является анализ взаимодействия Tello EDU, миниатюрного образовательного беспилотного летательного аппарата, и Turtlebot3, многофункционального беспилотного наземного аппарата [1, 2], в условиях реализации задач в помещениях с низким уровнем радиосигнала. При этом особое внимание уделяется применению локальной сети и операционной системы роботов

(ROS) как основных инструментов для управления и координации этих двух современных робототехнических систем.

На практике подобные условия могут возникнуть в многочисленных сценариях, в частности, в больших зданиях, включая склады, медицинские учреждения или офисные комплексы. В этих местах беспилотные аппараты могут выполнять самые разные задачи – от инвентаризации и управления поставками до мониторинга состояния окружающей среды.

Целью данного исследования является анализ особенностей применения ROS в качестве основного инструмента для контроля и координации роботов. В ходе работы будут подробно исследованы конкретные примеры использования и наиболее удачные модели реализации координации действий Tello EDU и Turtlebot3 при использовании операционной системы ROS [3].

Данная статья представляет определённый интерес для учёных и специалистов в области робототехники, так как позволяет углубить знания о применении ROS и локальных сетей в условиях слабого сигнала. В ней представлены практические рекомендации и решения, разработанные в ходе научных исследований и экспериментов, которые могут оптимизировать работу роботизированных систем и улучшить качество проведения их задач в специфических условиях.

Общий обзор проблемы слабого сигнала

Проблема слабого сигнала – это широко распространенная проблема в области беспроводных коммуникаций, особенно в помещениях. Она возникает из-за множества факторов, включая препятствия на пути сигнала (такие как стены или мебель), интерференцию от других беспроводных устройств или сетей, а также ограниченную мощность передачи устройства.

Слабый сигнал может вызвать различные проблемы, включая снижение производительности и скорости передачи данных, увеличение времени задержки и потерю пакетов данных. В крайних случаях это может привести к полному обрыву связи между устройствами.

В контексте взаимодействия Tello EDU и TurtleBot3 эта проблема может стать серьезным препятствием. Оба этих аппарата [4, 5] в значительной степени основаны на беспроводных коммуникациях для управления и передачи данных. Если сигнал слаб, это может негативно сказаться на их способности к взаимодействию и выполнению своих функций. Для исследования и разработок в области робототехники важно разработать эффективные стратегии преодоления проблемы слабого сигнала.

Использование локальной сети для решения проблемы

Применение локальной сети для взаимодействия между Tello Edu и Turtlebot3 может представлять собой эффективное решение проблемы слабого сигнала. В таком случае, вместо прямого беспроводного соединения между двумя устройствами, оба они соединяются с одной локальной сетью (часто через точку доступа Wi-Fi), позволяющей им обмениваться данными.

Несмотря на то, что препятствия и интерференции все еще могут вызывать проблемы со слабым сигналом, локальная сеть обычно обеспечивает более стабильное и надежное соединение по сравнению с прямыми беспроводными соединениями.

Радиус действия сигнала точки доступа обычно больше, чем у большинства беспилотных устройств, что обеспечивает более широкий диапазон работы. Кроме того, в большинстве современных сетей используются технологии, такие, как MIMO (Multiple Input Multiple Output), для улучшения производительности в условиях слабого сигнала или высокой плотности устройств.

Очень важно упомянуть, что и Turtlebot3, и Tello EDU уже предназначены для работы в Wi-Fi сетях, что облегчает интеграцию их с локальными сетями. На рис. 1 представлены данные беспилотные аппараты.

а) б)

Рис. 1. – Беспилотники: DJI Tello Edu (а) и Turtlebot3 burger (б)

Вместе с ROS (Robot Operating System), который предлагает инструменты и библиотеки для управления роботами и обмена информацией, применение локальной сети может значительно повысить эффективность взаимодействия двух устройств [6, 7].

Преимущества использования локальной сети:

1. Большая скорость передачи данных: LAN обеспечивает, как правило, более высокую пропускную способность, что может быть полезным для обмена большими объемами данных.

2. Больший диапазон: Local Area Networks (LAN), особенно те, которые работают через Wi-Fi, как правило, предлагают больший диапазон, чем прямое беспроводное соединение между двумя устройствами.

3. Надежность: если ассоциированный с LAN Wi-Fi маршрутизатор упадет или откажет, вся сеть все равно продолжит работать. Это не так с прямыми беспроводными соединениями, где отказ одного устройства может привести к потере связи.

Недостатки использования локальной сети:

1. Зависимость от инфраструктуры: для создания LAN необходим доступ к инфраструктуре сети, такой как Wi-Fi маршрутизатор.

2. Повышенная сложность: организация LAN может потребовать определенных технических знаний и настроек.

3. Интерференция: и телефоны, и другие устройства также используют Wi-Fi, что может создавать интерференцию и приводить к ухудшению производительности.

Причина, по которой LAN может быть лучшим решением, связана с преимуществами, упомянутыми выше, а именно, больший диапазон, большая скорость передачи данных и надежность. Оба устройства (Tello EDU и TurtleBot3) поддерживают Wi-Fi, что упрощает их подключение к LAN. Кроме того, при использовании ROS для управления обоими устройствами, LAN обеспечивает стабильную и надежную среду для передачи сообщений и данных между устройствами.

Переход на ROS и важность ее работы

Robot Operating System (ROS) является гибкий фреймворк, который служит для создания программного обеспечения для роботов. Она включает в себя набор библиотек, инструментов и соглашений, при помощи которых создание сложных и надежных робототехнических систем упрощается. Таким образом, ROS играет важную роль во взаимодействии между Tello EDU и Turtlebot3, так как она обеспечивает единый механизм для обмена данными между различными устройствами и системами.

В основе ROS лежит понятие "узлов" (nodes), которые являются процессами, выполняющимися в оборудовании робота. Узлы могут публиковать сообщения в определенные темы (topics), а другие узлы могут подписываться на эти темы для получения обновлений. Это позволяет

устройствам, таким, как Tello EDU и Turtlebot3, обмениваться данными посредством публикации и подписки на сообщения.

Так, например, Tello EDU может публиковать сообщения о своем текущем положении, а Turtlebot3 – подписываться на эти сообщения и обрабатывать их для выполнения своих действий, таких как движение или обнаружение препятствий. Эта архитектура делает ROS идеальной для синхронизации и координации действий различных устройств в робототехнической системе.

Использование ROS также обеспечивает дополнительное преимущество в виде доступа к широкому спектру уже разработанных инструментов и пакетов, которые могут быть использованы для ускорения разработки и облегчения интеграции различных устройств.

ROS (Robot Operating System) имеет несколько функций, которые делают его особенно подходящим для работы в помещениях с плохим сигналом [8, 9].

1. Оффлайн-работа: ROS позволяет роботам работать автономно, что означает, что они могут продолжать выполнять задачи, даже если сигнал сети слабый или отсутствует. Данные с сенсоров могут быть собраны и сохранены для последующего анализа, когда сеть снова станет доступной.

2. Публикация и подписка на сообщения: несмотря на проблемы с сетью, ROS все равно позволяет роботам общаться между собой через механизм публикации и подписки на сообщения. Это означает, что если один робот может получить какую-то информацию, он может разделить ее со всеми остальными, даже если они не могут напрямую связаться с источником данных.

3. Сетевая отказоустойчивость: ROS организован таким образом, чтобы поддерживать отказоустойчивость сети. Это означает, что если сетевое соединение обрывается или становится недоступным, система автоматически

переключится на использование других доступных соединений, позволяя роботам продолжать работать.

4. Использование набора данных SLAM: с целью внутренней навигации роботы могут использовать функционал ROS для создания карты окружающей среды при помощи технологии одновременной локализации и картографирования (SLAM) [10]. Это позволяет им уклоняться от препятствий и навигировать даже в условиях слабого сигнала.

5. Разнообразие поддерживаемых сенсоров: ROS поддерживает расширенный набор сенсоров, которые могут помочь в условиях слабого или отсутствующего сигнала. Например, ультразвуковые датчики, инфракрасные датчики и LIDAR могут быть использованы для сбора данных об окружающей среде.

Однако важно отметить, что, хотя ROS имеет средства для работы в условиях слабого сигнала, ни одна система не может полностью компенсировать проблемы, вызванные потерей сетевого соединения. Для успешной работы робота его системы и оборудование должны быть правильно сконфигурированы и синхронизированы.

Реализация работы

Чтобы мы могли взаимодействовать с Tello Edu и Turtlebot3, необходимо в python установить следующие библиотеки:

djitellopy – библиотека, позволяющая давать команды Tello Edu на языке Python;

rospy – библиотека, которая может взаимодействовать со всеми компонентами ROS;

cv2 – библиотека компьютерного зрения, предназначенная для обработки изображений;

geometry_msgs – эта библиотека предоставляет сообщения для распространенных геометрических примитивов, таких, как точки, векторы и положения;

std_msgs – эта библиотека предоставляет сообщения в виде команд;

cv_bridge – это пакет для преобразования между ROS и OpenCV и форматами изображений;

move_base_msgs.msg – подкаталог узлов, позволяющий дрону двигаться по квадрату;

sensor_msgs.msg – этот пакет предоставляет множество сообщений и служб, относящихся к сенсорным устройствам;

actionlib – Пакет actionlib предоставляет инструменты для создания серверов, выполняющих долгосрочные задачи, которые могут быть вытеснены. Он также предоставляет клиентский интерфейс для отправки запросов на сервер;

tf (TensorFlow) – библиотека, предназначенная для создания нейронной сети.

На рис. 2 показан принцип работы взаимодействия двух видов беспилотных аппаратов.

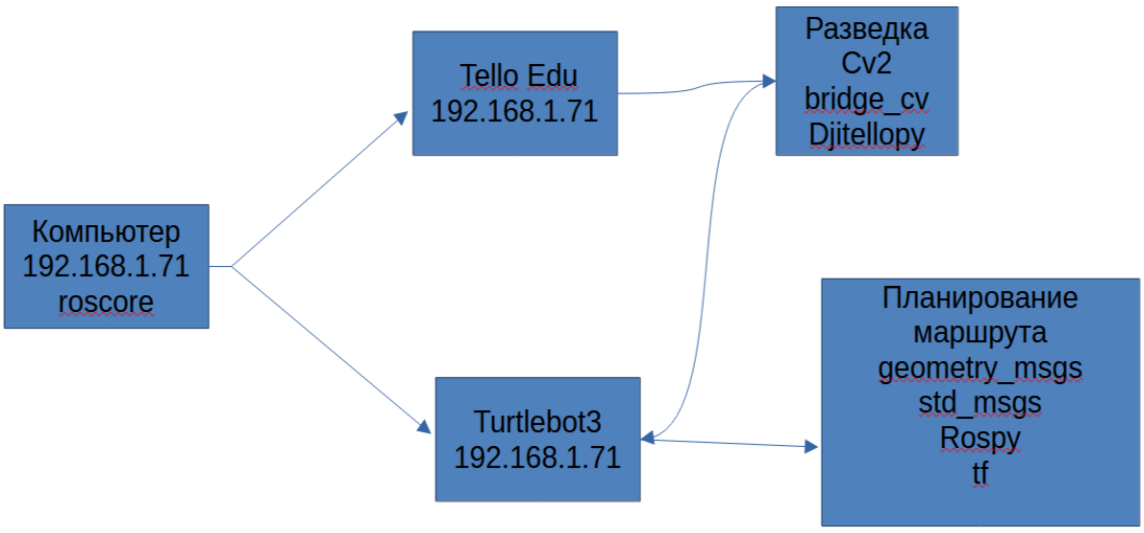

Рис. 2. – Схема взаимодействия Tello Edu и Turtlebot3

Компьютер и беспилотники подсоединяются к одному IP-адресу. После подключения всех беспилотников, начинает работу Tello Edu. Он разведывает территорию и при помощи библиотеки Cv2 отмечает определенные объекты и запоминает их. Далее он опубликовывает сообщение об изображении в теме изображений, чтобы его могли получить другие узлы ROS. Потом обработанные изображения он отправляет Turtlebot3 для построение карты с использованием технологии SLAM. И при всех обработанных изображении он выстраивает оптимальный маршрут. Пакет Geometry_msgs показывает координаты передвижения Turtlebot3.

Ниже представлен код взаимодействия Tello Edu и Turtlebot3.

```
# Код для Tello Edu
import cv2
import rospy
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal
from cv_bridge import CvBridge
from djitellopy import Tello
from sensor_msgs.msg import Image
import actionlib
import tf
from Tello_Edu import bridge
from geometry_msgs.msg import Point     # сообщение с позициями
from std_msgs.msg import String          # сообщение с командами
# Предполагается, что у вас уже есть созданный объект Tello, который подключен к Tello EDU
tello = Tello()
def capture():
    # Создание объекта для захвата видео
    cap = cv2.VideoCapture(0)
    # Начать поток видео
    tello.streamon()
    # Переместить Tello по заранее заданной траектории
    navigate()
    # Захват видео
    for frame in tello.get_frames():
        # Преобразование кадра в сообщение об изображении ROS
        image_message = bridge.cv2_to_imgmsg(frame, 'bgr8')
```

```
        # Опубликовать сообщение об изображении в теме изображений, чтобы его могли получить другие узлы ROS
        pub = rospy.Publisher('tello/image', Image, queue_size=10)
        pub.publish(image_message)
def navigate():
    tello.takeoff()
    # Добавьте здесь вашу логику навигации, например: tello.move_left, tello.move_up, ...
    tello.land()
if __name__ == '__main__':
    rospy.init_node('tello_navigation')
    capture()
# Turtlebot3 code
class TurtleBot3Controller:
    def __init__(self):
        # Инициализация узла
        rospy.init_node('turtlebot3_controller')
        self.bridge = CvBridge()
        # Подпись на тему изображения tello
        rospy.Subscriber('tello/image', Image, self.process_image)
        # Создать клиент MoveBase
        self.client = actionlib.SimpleActionClient('move_base', MoveBaseAction)
        self.client.wait_for_server()
    def process_image(self, image_msg):
        # Преобразование сообщения об изображении ROS в изображение OpenCV
        cv_image = self.bridge.imgmsg_to_cv2(image_msg, "bgr8")
        # Это где вы бы обрабатывали изображение для создания карты и пути, возможно, с использованием SLAM
        self.navigate_to_goal()
    def navigate_to_goal(self):
        while not rospy.is_shutdown():
            goal_pose = self.create_goal_from_path()
            goal = MoveBaseGoal()
            goal.target_pose.header.frame_id = "map"
            goal.target_pose.header.stamp = rospy.Time.now()
            goal.target_pose.pose.position.x = goal_pose['x']
            goal.target_pose.pose.position.y = goal_pose['y']
            quaternion = tf.transformations.quaternion_from_euler(0, 0, goal_pose['yaw'])
            goal.target_pose.pose.orientation.x = quaternion[0]
            goal.target_pose.pose.orientation.y = quaternion[1]
```

```
            goal.target_pose.pose.orientation.z = quaternion[2]
            goal.target_pose.pose.orientation.w = quaternion[3]
            self.client.send_goal_and_wait(goal)
    def create_goal_from_path(self):
        # Это где вы бы создали цель из пути
        return {'x': 0.0, 'y': 0.0, 'yaw': 0.0}
# Функции обратного вызова, которые вызываются при получении нового сообщения
def tello_callback(data):
    # обработка данных
    # ...
    return
def turtlebot_callback(data):
    # обработка данных
    # ...
    return
def manager_node():
    rospy.init_node('manager_node')
    # подписываемся на топики с позициями
    rospy.Subscriber("/Tello_Edu/position", Point, tello_callback)
    rospy.Subscriber("/Turtlebot3/position", Point, turtlebot_callback)
    # публикация команд
    tello_command_pub = rospy.Publisher('/Tello_Edu/command', String, queue_size=10)
    turtlebot_command_pub = rospy.Publisher('/Turtlebot3/command', String, queue_size=10)
    rate = rospy.Rate(10) # 10 Hz
    while not rospy.is_shutdown():
        # здесь можно добавить скрипты для обработки и создания команд, например:
        tello_command = String(data="command for Tello Edu")
        turtlebot_command = String(data="command for Turtlebot3")
        # публикация команд
        tello_command_pub.publish(tello_command)
        turtlebot_command_pub.publish(turtlebot_command)
        rate.sleep()
if __name__ == '__main__':
    controller = TurtleBot3Controller()
    rospy.spin()
    manager_node()
```

Первая часть программы осуществляет разведку территории с помощью Tello Edu. Робот будет летать по заданной траектории, записывать видео и передавать его в реальном времени.

Вторая часть программы относится к Turtlebot3, который принимает видеопоток, преобразует его в карту с помощью алгоритма SLAM (Simultaneous Localization and Mapping), затем планирует маршрут на основе этой карты и начинает движение.

Заключение

Для запуска данного алгоритма требуется Tello Edu и Turtlebot3, так как при помощи библиотеки “cv” Tello Edu записывает на свою камеру видео, затем видео обрабатывает и передает Turtlebot3, тем самым он прорабатывает данный материал, преобразует в карту и по технологии SLAM выстраивает по обработанной карте оптимальный маршрут.

Литература

1. Хорсик И.А., Бунина Л.В. Уязвимости и методы защиты операционной системы ROS при реализации мультиагентной системы на базе робота Turtlebot3 // Инженерный вестник Дона, 2023, №12. URL: ivdon.ru/ru/magazine/archive/n12y2023/8862.

2. Габдуллин А.Р. Буйвал А.К. Лавренов Р.О. Магид Е.А. ROS-моделирование взаимодействия бпла и наземного беспилотного робота для решения задачи планирования маршрута в статической среде // III Всероссийский научно-практический семинар «Беспилотные транспортные средства с элементами искусственного интеллекта» (БТС-ИИ-2016). Иннополис, 2016. С. 21-30.

3. Dieber Bernhard, Breiling Benjamin, Taurer Sebastian, Kacianka Severin, Rass Stefan, Schartner Peter Security for the Robot Operating System // Robotics and Autonomous Systems, 2017. pp. 1-29.

4. Костюков В.А., Гисцов В.Г. Евдокимов И.Д. Особенности учета кинематических ограничений при планировании траектории БпЛА // Инженерный вестник Дона, 2023, №11 URL: ivdon.ru/ru/magazine/archive/n11y2023/8833.

5. Buniyamin N. et al. A simple local path planning algorithm for autonomous mobile robots // International journal of systems applications, Engineering & development. 2011. V. 5. №. 2. pp. 151-159.

6. Лазарев В.С., Лащев А.А. Разработка математической модели БПЛА на базе квадрокоптера с рамой DJI F-450 // Инженерный вестник Дона, 2018, №2 URL: ivdon.ru/ru/magazine/archive/N2y2018/5001.

7. Melchior, Nik A., and Reid Simmons. Particle RRT for path planning with uncertainty. // IEEE Robotics and Automation, 2007. pp. 1617-162.

8. Марченко Е.П. Основные понятия о robot operating system и ее компонентах // Вопросы науки и образования 2017. С. 36-41.

9. Quigley Morgan, Gerkey Brian, Smart William D. Programming Robots with ROS: A Practical Introduction to the Robot Operating System, 2015. p. 417.

10. Grimmett Richard // Raspberry Pi Robotic Projects, 2016. p. 238.

References

1. Horsik I.A., Bunina L.V. Inzhenernyj vestnik Dona, 2023, №12, URL: ivdon.ru/ru/magazine/archive/n12y2023/8862.

2. Gabdullin A.R., Bujval A.K., Lavrenov P.O., Magid E.A. III Vserossijskij nauchno-prakticheskij seminar "Bespilotnye transportnye sredstva s elementami iskusstvennogo intellekta" (BTS-II-2016), Innopolis, 2016. pp. 21-30.

3. Dieber Bernhard, Breiling Benjamin, Taurer Sebastian, Kacianka Severin, Rass Stefan, Schartner Peter Robotics and Autonomous Systems, 2017. pp. 1-29.

4. Kostyukov V.A., Giscov V.G. Evdokimov I.D. Inzhenernyj vestnik Dona, 2023, №11. URL: ivdon.ru/ru/magazine/archive/n11y2023/8833.

5. Buniyamin N. et al. International journal of systems applications, Engineering & development. 2011. V. 5. №. 2. pp. 151-159.

6. Lazarev V.S., Lashchev A.A. Inzhenernyj vestnik Dona, 2018, №2. URL: ivdon.ru/ru/magazine/archive/N2y2018/5001.

7. Melchior, Nik A., and Reid Simmons. Particle RRT for path planning with uncertainty. IEEE Robotics and Automation, 2007. pp. 1617-162.

8. Marchenko E.P. Voprosy nauki i obrazovaniya 2017. pp. 36-41.

9. Quigley Morgan, Gerkey Brian, Smart William D. Programming Robots with ROS: A Practical Introduction to the Robot Operating System, 2015. p. 417.

10. Grimmett Richard Raspberry Pi Robotic Projects, 2016. p. 238.

Дата поступления: 16.11.2023
Дата публикации: 3.01.2024