Создание танца нескольких квадрокоптеров с управлением из ROS

Что такое ROS

ROS — Операционная система для роботов — это фреймворк для программирования роботов, предоставляющий функционал для распределённой работы.

ROS обеспечивает стандартные службы операционной системы, такие как:

- аппаратную абстракцию,
- низкоуровневый контроль устройств,
- реализацию часто используемых функций,
- передачу сообщений между процессами,
- и управление пакетами.

ROS основан на архитектуре графов, где обработка данных происходит в узлах, которые могут получать и передавать сообщения между собой. Библиотека ориентирована на Unix-подобные системы (Ubuntu Linux включен в список «поддерживаемых» в то время как другие варианты, такие как Fedora и Mac OS X считаются «экспериментальными»).

ROS имеет две основные «стороны»: стороны операционной системы ros, как описано выше и ros-pkg, набор поддерживаемых пользователями пакетов (организованных в наборы, которые называются стек), которые реализуют различные функции робототехники: SLAM, планирование, восприятие, моделирование и др.

ROS выпускается в соответствии с условиями BSD-лицензии и с открытым исходным кодом. ROS бесплатен для использования, как в исследовательских, так и в коммерческих целях. Пакеты из ros-pkg распространяются на условиях различных открытых лицензий

Ardrone_autonomy является ROS драйвером для квадрокоптера Parrot ArDrone.

Поддерживает квадрокоптеры ArDrone 1.0 и ArDrone 2.0.

Этот пакет является ответвлением пакета ArDrone Brown.

Пакет позволяет:

- получать сообщения с датчиков ArDrone,
- получать изображения с камер,
- управлять движением квадрокоптера
- управлять свечением светодиодов.

Установка пакета

Переходим на страницу проекта https://github.com/AutonomyLab/ardrone_autonomy

Там описана установка пакета из исходников. Установка в рабочее пространство catkin

cd ~/catkin_ws/src

git clone https://github.com/AutonomyLab/ardrone_autonomy.git -b hydro-devel

cd ~/catkin_ws

catkin_make

Для запуска пакета

rosrun ardrone_autonomy ardrone_driver

или запуск командного файла в котором можно выставить настройки,

отличные от настроек по умолчанию

roslaunch ardrone_autonomy ardrone1.launch

При запуске пакета создается один узел

rosnode list

Для публикации сообщений, изображений пакет создает множество тем

rostopic list

И запускает несколько сервисов

rosservice list

Посмотрим выдаваемые квадрокоптером в тему /ardrone/navdata сообщения

rostopic echo /ardrone/navdata

Или картинку /ardrone/image_raw

Посмотреть тип сообщения

rostopic type ardrone/image_view

Посмотреть формат типа сообщения

rosmsg show [theme]

Для просмотра изображений публикуемых в темы ROS есть пакет image_view

rosrun image_view image_view image:=/ardrone/image_raw

rosrun image_view image_view image:=/ardrone/front/image_raw

В ardrone используется 2 камеры

rosrun image_view image_view image:=/ardrone/bottom/image_raw

Для переключения на другую камеру Вызов сервиса /ardrone/togglecam

rosservice call /ardrone/togglecam

Теперь можно посмотреть и изображение нижней камеры

rosrun image_view image_view image:=/ardrone/bottom/image_raw

rosrun image_view image_view image:=/ardrone/image_raw

Отправка команд для AR-Drone

Взлет — отправка пустого сообщения в тему ardrone/takeoff

Посадка - отправка пустого сообщения в тему ardrone/land

Сборос параметров(аварийная остановка) - отправка пустого сообщения в тему ardrone/reset

rostopic pub /ardrone/takeoff std_msgs/Empty

rostopic pub /ardrone/land std_msgs/Empty

После взлета для управления движением ArDrone необходимо посылать сообщения типа geometry_msgs::Twist в тему cmd_vel

-Linear.x: двигаться назад

+ Linear.x: двигаться вперед

-Linear.y: : движение вправо

+ Linear.y: движение влево

-Linear.z: двигаться вниз

+ Linear.z: двигаться вверх

-Angular.z: повернуть налево

+ Angular.z: повернуть направо

Диапазон для каждого компонента должно быть от -1,0 до 1,0. Максимальный диапазон может быть настроен с помощью ROS параметры

Светодиодные анимации

Вызов службы ardrone/setledanimation будет вызывать выполнение одной из 14 предопределенных светодиодной анимаций для ArDrone.

Параметры

- uint8 типов : тип анимации, который является число в диапазоне [0 .. 13];
- float32 частоты : частота анимации в Гц;

- uint8 продолжительность : продолжительность анимации в секундах.

rosservice call /ardrone/setledanimation 1 4 5

Полетные анимации

Вызов службы ardrone/setflightanimation будет выполнять одну из 20 предопределенных полетных анимаций (полетных фигур) для ArDrone. Параметры:

- uint8 типов : тип полета анимация, число в диапазоне [0 .. 19]
- uint16 продолжительность : продолжительность анимации. Используйте 0 для длительности по умолчанию (рекомендуется)

rosservice call /ardrone/setflightanimation 1 0

Подключаем джойстик

Подключаем джойстик к компьютеру с Linux.

ls /dev/input

На сервере параметров устанавливаем параметр joy_node/dev, где указываем порт подключения нашего джойстика

rosparam set joy_node/dev "/dev/input/js0"

И запускаем узел joy_node из пакета joy

rosrun joy joy_node

И смотрим сообщения, публикуемые в тему joy

Для управления квадрокоптером надо получать сообщения из темы joy

Создать узел, который будет выступать в качестве слушетеля (subscriber) темы joy

И на основе полученных данных

- осуществлять взлет/посадку отправкой сообщений в темы /ardrone/takeoff, /ardrone/land

- управлять квадрокоптером отправкой сообщений в тему cmd_vel

- управлять светодиодами вывовом сервиса /ardrone/setledanimation

- запускать полетные анимации вызовом сервиса /ardrone/setledanimation

Создаем скрипт на python – ros_ardrone2_joystick_to_move1.py

Создание танца квадрокоптеров

Наша задача создать танец квадрокоптеров –

И нужно табло – web-интерфейс для запуска и управления квадрокоптерами,

Находящимися на разных компьютерах.

Будем использовать пакет rosbridge

Создан в Университет Брауна, rosbridge был первоначально предназначен для любого не ROS процессов клиента для работы с системой ROS. Rosbridge позволяет внешним клиентам иметь доступ к темам и сервисам ROS (публикация и получение из тем, вызов сервисов). Rosbridge является частью мете-пакета rosbridge_suite, включающего различные дополнительные пакеты для реализации протокола rosbridge.

Установка пакета

sudo apt-get install ros-hydro-rosbridge-server

sudo apt-get install ros-hydro-rosbridge-suite

Программа писалась год назад и веб-интерфейсе отображался вид с камеры. Поэтому устанавливался пакет mjpeg-server.

sudo apt-get install ros-hydro-mjpeg-server

К сожалению новые версии Google Chrome не поддерживают отображение потокового видео

комп очень сильно зависает, поэтому из web-интерфейса убрана камера

Rosbridge клиенты могут быть написаны на любом языке, который поддерживает WebSockets. Библиотеки с открытым источником WebSocket клиента доступны для Python, Java, Android, C и пр..

Мы будем использовать Javascript

И нам нужен на одном из компьютеров, где находится табло, установить web-сервер

Будем использовать пакет roswww_pkg

cd ~/catkin_ws/src

git clone https://github.com/tork-a/roswww_pkg.git

cd ~/catkin_ws

catkin_make

И запуск

rosrun rosbridge_server rosbridge_websocket

rosrun mjpeg_server mjpeg_server

rosrun roswww webserver.py

Для использования в блоке html-страницы подключаем библиотеку ros.js

<script type="text/javascript" src="ros.js"></script>

Создаем websocket соединение к rosbridge по порту 9090

var con = new Bridge("ws://ip_robot:9090");

Теперь можно получать и публиковать сообщения в ROS

Публикация сообщения в тему — con.publish(topic, msg), например

con.publish('/cmd_vel', {"linear":{"x":1.0,"y":0,"z":0},"angular":{"x":0,"y":0,"z":0}});

con.publish('/cmd_vel', {"linear":{"x":1.0,"y":0,"z":0},"angular":{"x":0,"y":0,"z":0}});

Получение сообщения от сервиса

con1.callService(cback11,'/vp_ardrone2_get_scenario',outarr);

Если тема не существует — до отправки сообщений необходимо создать тему — con.advertise(topic, type), например

con1.advertise('/web_publisher', 'std_msgs/String');

Чтобы подписаться на получение сообщений из темы — сначала создаем функцию обратного вызова, а затем подписаться на получение сообщений из темы, например:

var get_data1(msg1) {alert(msg1);}

var cback1=function(msg1){JSON.stringify(msg1);get_data1(msg1);}con.subscribe(cback1,"/from_arduino1","stg_msgs/String");

var cback2=function(msg2) {JSON.stringify(msg2),get_ros(11,msg2);}
con1.subscribe(cback2,'/web_subscriber','std_msgs/String');

На табло

чекбоксы для выбора коптеров

кнопка запуска

плеер

для проверки кнопки Запуск всех (выбранных)

Приземление всех (выбранных)

статус коптера

заряд батареи

высота

поворот по Z

Отклонение по осям x, y при движении по z

Используем библиотеку для компьютерного зрения opencv

ROS использует изображения в своем собственном формате фозрмата сообщения, но мноИгие пользователи захотят использensor_msgs/Image. Для использования изображения с opencv необходимо производить конвертацию изображения в формат opencv. Используется ROS библиотека CvBridge. Это библиотека ROS, что обеспечивает интерфейс между ROS и OpenCV.

Вот пример

```
image_sub = rospy.Subscriber("ardrone/image_raw",Image,self.callback)
```

```
 def callback(data):
   try:
     cv_image = bridge.imgmsg_to_cv(data, "bgr8")
   except CvBridgeError, e:
     print e
 image_pub = rospy.Publisher("image_converter1",Image)
```

….....

```
image_pub.publish(self.bridge.cv_to_imgmsg(img_g1, "mono8"))
```

Ссылки

Русскоязычная документация по ROS - http://robocraft.ru/page/robotics/

Работа с Rosbridge (пример отправки.получения из web-интерфейса) – http://robocraft.ru/blog/2945.html

Управление квадрокоптером с помощью голоса - http://robocraft.ru/blog/2882.html

Управление квадрокоптером Ardrone с помощью джойстика - http://cxem.net/uprav/uprav53.php

Видео работы с интерфейсом Танец квадрокоптера – http://www.youtube.com/watch?v=NihKr0LbHCE

http://www.youtube.com/watch?v=QTXNHLGDwnI