DJI Onboard API Linux ROS Sample

说明文档

版本	时间	描述
V1.0.0	2015-05	创建
V1.0.1	2015-06	修改目录

文档介绍基于 ROS(Robot Operating System)的 DJI Onboard API C++例程 ，该例程使用了 ROS 软件包“keyboardteleop.js”，编译该程序包后，通过 HTML 网页 GUI 进行飞机的基本控制，如起飞、降落、返航等。

开发环境

主机平台：Ubuntu12.04、Ubuntu14.04
ROS 包：ROS hydro、ROS indigo
浏览器：Firefox

例程目录结构

例程：DJI_Onboard_API_ROS_Sample

目录	说明
dji_sdk	子目录 src：包含例程源代码。 子目录 launch：包含 ros 包的 launch 文件
dji_keyboard_ctrl	目录中 sdk_keyboard_demo.html 为 GUI 网页
doc	文档

ROS 安装

开发者参考以下 ROS wiki 安装 ROS 到 Linux 主机中。
http://wiki.ros.org/ROS/Installation
安装完后，安装 rosbridge server 包。Ubuntu 下通过如下形式命令安装：
sudo apt-get install ros-[ROS VERSION]-rosbridge-server
如果开发者安装的 ROS 版本为 hydro，则安装 rosbridge server 包的命令为：

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

主要功能函数

- 串口配置

int Pro_Hw_Setup(QString port_name, int baudrate)

函数功能：配置并打开 Linux 下串口。
函数参数：**port_name** 串口设备文件名，baudrate 串口波特率。
函数返回值：1 成功；0 失败。

- API 激活函数

void ros_activation_callback(const std_msgs::Float32::ConstPtr& msg)

函数功能：激活 API。
函数参数：ROS Float32 消息。
函数返回值：无。

- 获取或释放飞机控制权

void ros_nav_open_close_callback(const std_msgs::Float32::ConstPtr& msg)

函数功能：激活 API 后，获取或释放控制权
函数参数：ROS Float32 消息，消息数据为 1 请求获取控制权；0 释放控制权
函数返回值：无。

- 基本的飞行控制

void ros_cmd_data_callback(const std_msgs::Float32::ConstPtr& msg)

函数功能：基本的起飞、降落、返航控制。
函数参数：ROS Float32 消息，消息数据为 1 返航、4 起飞、6 降落。
函数返回值：无。

例程配置

例程配置前，开发者需要通过 DJI 网站注册获得 APP id、API level 以及密钥。

编辑 *dji_sdk/launch/sdk_demo.launch*，如下图所示，根据获得 APP id、API level、密钥 key，以及使用的串口设备名和波特率修改对应项目。

```
<launch>
        <node pkg="dji_sdk" type="dji_sdk_node" name="dji_sdk_node" output="screen">
                <!-- node parameters -->
                <param name="serial_name" type="string" value="/dev/ttyUSB0"/>
                <param name="baud_rate" type="int" value="230400"/>
                <param name="app_id" type="int" value="10086"/>
                <param name="app_api_level" type="int" value="2"/>
                <param name="app_version" type="int" value="1"/>
                <param name="app_bundle_id" type="string" value="12345678901234567890123456789012"/>
                <param name="enc_key" type="string" value="DJI-DEMO AES256 KEY-lala-haha-MA"/>
        </node>
</launch>
```

device name
baud rate
APP id
API level
Key

例程编译

将 dji_sdk 拷贝到 ROS workspace 下，使用 ros 编译命令 *catkin_make* 编译。

例程运行

编辑 *dji_keyboard_ctrl2/*sdk_keyboard_demo.html，把 url 中的地址改成当前 Linux 主机名或者 localhost（127.0.0.1），如下图所示

```
function init() {
// Connecting to ROS.
var ros = new ROSLIB.Ros({
url : 'ws://127.0.0.1:9090'
});
```

由于程序会访问 Linux 下的串口设备，开发者在运行程序前需要保证程序具有访问该串口的权限。假设串口设备名为/dev/ttyUSB0，在 Ubuntu Linux 下可以通过以下命令赋予程序访问串口的权限。

sudo chmod 777 /dev/ttyUSB0

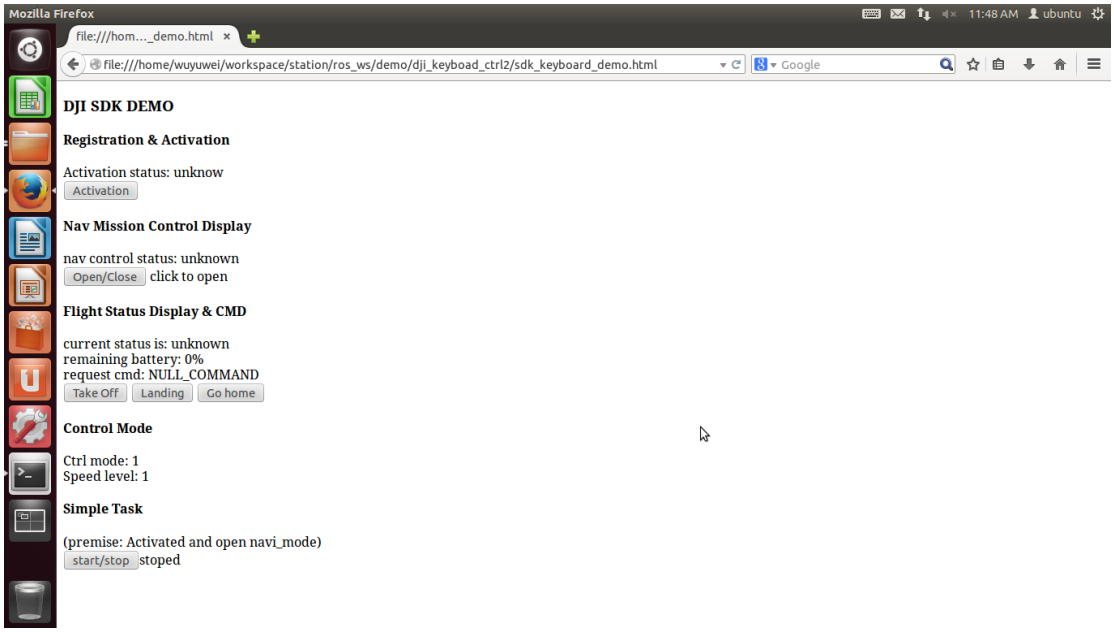
启动 rosbridge server。

roslaunch rosbridge_server rosbridge_websocket.launch

启动例程 launch。

roslaunch dji_sdk sdk_demo.launch

在浏览器中打开 *dji_keyboard_ctrl2/*sdk_keyboard_demo.html，如下图所示：



飞行控制

进行程序飞行控制前，通过串口线连接飞机和 Linux 主机，并设置飞机在 API 控制模式下。

点击“Activation”按钮激活 API，

点击“Open/Close”请求打开或关闭 API 控制模式。

点击“Take off”按钮，请求飞机起飞。

点击“Landing”按钮，请求飞机降落。

点击“Go Home”按钮，请求飞机返回 Home 点。

点击“Start/Stop” 按钮，飞机会完成一系列的简单命令，例如起飞，pitch、roll、yaw 控制和降落控制。为了飞行安全，开发者使用这个功能做真实飞行时，要确保飞机处在一个空阔的区域。

此外，通过按下 Linux 主机键盘“WASD”键控制飞行前后左右方向，“ZC” 控制竖直速度、“QE”控制偏航旋转。

“WASD”控制前后左右方向的倾角度数为 5*speed_level，这里 speed_level 默认为 1，通过键盘数字“123456”来修改，speed_level 修改后，姿态控制指令的数值也会随之改变。请谨慎使用大姿态角。