DJI Onboard API Sample for ROS

Version	Date	Remarks
V1.0.0	2015-05	Created

This document introduce the general features of the DJI Onboard API C++ sample based on Robot Operating System in Linux environment. Compile and run this sample to gain basic control of the aircraft such as, take-off, landing and go home by using the keyboardteleop.js library from the ROS.

Development Environment

Linux Distribution：Ubuntu12.04, Ubuntu14.04
ROS package: ROS hydro, ROS indigo
Internet Browser：Mozilla Firefox

Directory Structure

Directory structure for *DJI_Onboard_API_ROS_Sample* is listed as below:

Directory	Description
dji_sdk	Sub directory src: source code files Sub directory launch: ROS launch file
dji_keyboard_ctrl2	Use the sdk_keyboard_demo.html file as the HTML control panel
doc	Documents

ROS Installation

Refer to the ROS wiki link below on how to install the ROS to the Linux host.
http://wiki.ros.org/ROS/Installation
Install rosbridge server package when ROS has been successfully installed on to the Linux host. Use the command shown below to install the package into the Ubuntu:
sudo apt-get install ros-[ROS VERSION]-rosbridge-server
Use the command shown below to install the package into the Ubuntu, if you are using the ROS hydro:

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

List of key functions

- Serial Port Configuration

```
int Pro_Hw_Setup(QString port_name, int baudrate)
```

Usage：Setup and open serial port.
Parameters：`port_name` denotes the port number of the serial device. `baudrate` denotes transmission rate of the serial port.
Return Values：0 as Success and -1 as Failed.

- API Activation

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

Usage: Activate the DJI Onboard API.

Parameters: ROS Float32 message.

Return Value: Void.

- Gain or Release the Control of the Aircraft

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

Usage: Obtain or release the access to the aircraft after DJI OnBoard API is activated.

Parameters: ROS Float32 message. Value 1 denotes requesting control. Value 2 denotes releasing control.

Return Value: Void.

- Basic Aircraft Control

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

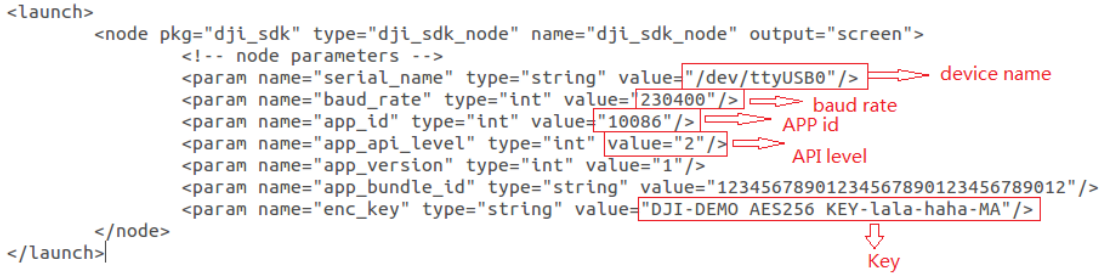
Usage: Allow aircraft to take off, land and return to home.

Parameters: ROS Float32 message. Value 1 denotes return to home control. Value 4 denotes take off. Value 6 denotes landing.

Return Value: Void.

Configuration

Developers must obtain the app ID, API level and encryption key, serial device name and the baudrate of the UART serial device before compiling the source files of the sample code. Open the launch file that located in dji_sdk/launch/ directory and input the values mentioned above.




```
<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>
```

Developers must ensure the baudrate set is consistent with the one of the aircraft.

Compile

Copy *dji_sdk* folder to the ROS workspace. Use the ROS compiling command *catkin_make* to start compiling.

Run

Edit file dji_keyboard_ctrl2/sdk_keyboard_demo.html and replace the URL with the localhost IP (127.0.0.1) as the screenshot shown below:

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

Ensure that the current account has access privilege to the serial device. Assume that the serial device is named as "/dev/ttyUSB0", use the following command to grand access privilege for the serial device.

sudo chmod 777 /dev/ttyUSB0

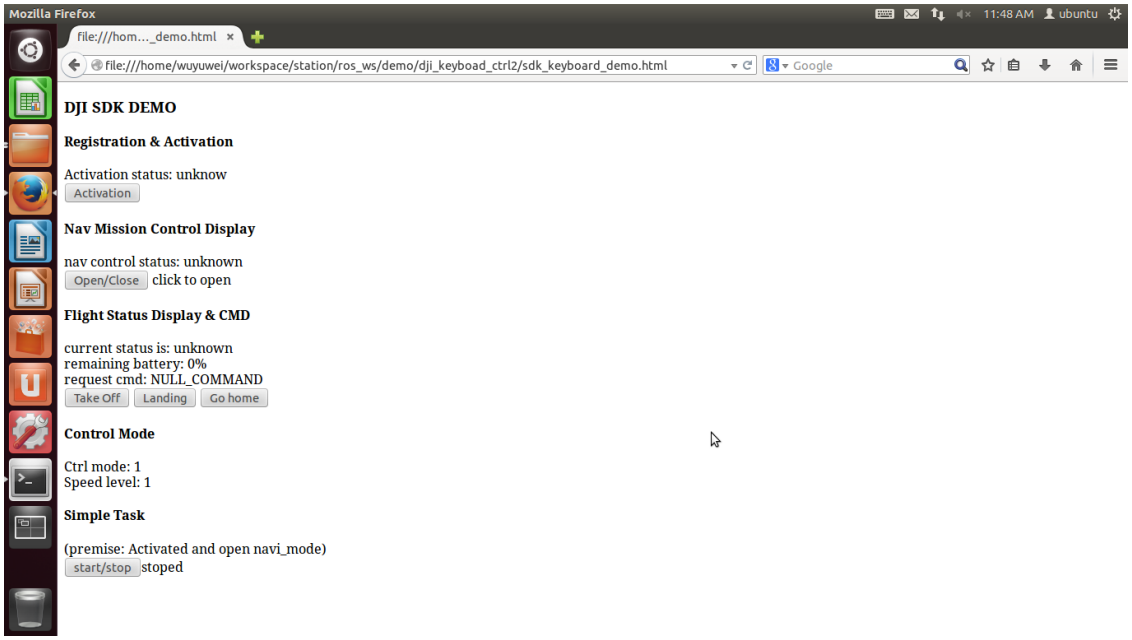
Launch rosbridge server.

roslaunch rosbridge_server rosbridge_websocket.launch

Run the launch demo.

roslaunch dji_sdk sdk_demo.launch

Open *dji_keyboard_ctrl2/sdk_keyboard_demo.html* by using web browse.



Aircraft Control

Connect the aircraft to the PC via a serial cable. Set the aircraft in API mode by switching ~~the current flight mode to A-mode using the remote controller. If the aircraft is in A-mode, then switch to either P-mode or F-mode and then switch it back to A-mode to set the aircraft in API mode.~~flight mode using remote controller.

Click "Activation" button to activate API，

Click "Open/Close" to enter or exit API mode.

Click "Take off" to send take off command to the aircraft.

Click "Landing" to send landing command to the aircraft.

Click "Go Home" to have the aircraft returned to the last recorded home point.

Click “Go Home” to have the aircraft performed a series combination commands such as takeoff, pitch, roll and landing. Attention for flight safety. Make sure you are in a wide area if you are not using simulator.

In addition, you may use “W”, “A”, “S”, “D” key on the keyboard to have the aircraft to move horizontally, “Z”, “C” to change the vertical velocity, and “Q”, “E” to change yaw.

The horizontal movement is controlled by angle command associated with button “W”, “A”, “S”, “D”. The angle is 5*speed_level, speed_level is an inner variable with default value 1. Its value can be changed by keyboard button “1”, “2”, “3”, “4”, “5”, “6”. Press different buttons can change the valule of angle command. Please be careful when you are using large angle commands. Aircraft will be accelerate rapidly.