**https://www.theconstructsim.com/start-programming-drones-using-ros-video-answer/**

**Step1. Create a new project on ROS Development Studio(RDS)**

We’ll use ROS Development Studio(RDS) for this tutorial, you can register a free account [here](https://www.theconstructsim.com/rds-ros-development-studio/?next=https%3A//rds.theconstructsim.com/simulations/).

After logging into RDS, click on create my project. It will move to the public simulation. You can find tons of public simulation here offered by the construct for free and start to work on any of them in just minutes. See how powerful RDS is! For today, we’ll use sjtu\_drone\_tc project. Please click on it. Click the tools menu, you can find some tools that help you develop in RDS. For example, you have the:

1. Shell: it is the terminal where you can execute commands in RDS. You can open it as many as you want in RDS!
2. IDE: It’s the best way to explore the source tree of your project. With a right click, you can add or remove files easily.
3. Jupyter Notebook: You can take notes for your project here. Since it’s working with python shell, you can directly execute python script here. A default notebook is provided to help you start the simulation.
4. Graphical tool: You can use all the GUIs supported by ROS here(e.g. RViz, rqt\_gui…etc.)

**Notice**:

We are not automatically running the simulation when you start RDS now. In order to have the same simulation shown in the video. Please go **Simulations->Select launch file->main.launch**to launch it by yourself. Then you can type the following command in a shell to check if the topics are correctly publishing by the drone.

|  |  |
| --- | --- |
| 1 | **$ rostopic list** |

**Step2. Get started with the simulation**

Let’s get started by following the instruction in the default jupyter notebook. Open it from **tools->jupyter notebook->default.ipynb**.

We can make the drone take off with the shell command

|  |  |
| --- | --- |
| 1 | $ rostopic pub /drone/takeoff std\_msgs/Empty "{}" |

You should see the drone take off as soon as you send this command.

**Notice:**

You can use the ROS auto-completion function while you are typing a ROS command by pressing [TAB]. It’s a good idea to do that when the command is too long and hard to type it correctly.

You can also land the drone with the following command

|  |  |
| --- | --- |
| 1 | $ rostopic pub /drone/land std\_msgs/Empty "{}" |

You can also find an instruction in the default.ipynb shows you how to do it with a python script instead of sending commend from shell.

**Step3. Program with drones**

We have more examples for you! Let’s say, we want to use the position control function provided by the drone. We found there is a topic called /drone/posctrl, but how to use it? By typing

|  |  |
| --- | --- |
| 1 | rostopic info /drone/posctrl |

You’ll see the output like this.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Type: std\_msgs/Bool    Publishers:  \* /my\_node (http://ip-172-31-35-31:45972/)    Subscribers:  \* /gazebo (http://10.8.0.1:44685/) |

It seems that the topic is using the Bool message, but what is Bool message and how can I use it? You can further investigate it by typing

|  |  |
| --- | --- |
| 1 | rosmsg show std\_msgs/Bool |

and got the output

|  |  |
| --- | --- |
| 1 | bool data |

The Bool message is very simple. It contains only one attribute called data with the type bool. Let’s try to send a message to this topic! Before we publish to the topic, we set up a monitor first with

|  |  |
| --- | --- |
| 1 | rostopic echo /drone/posctrl |

Then copy, paste and execute the following code in jupyter notebook.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | from std\_msgs.msg import Bool    var\_bool = Bool()  pub\_posctrl = rospy.Publisher('/drone/posctrl',Bool,queue\_size = 1)  var\_bool.data = True  pub\_posctrl.publish(var\_bool) |

You should see

|  |  |
| --- | --- |
| 1 | data: True |

which means the message is published correctly. We enable the position control function on the drone successfully. Similarly, you can move the drone by publishing Twist message to the /cmd\_vel topic. Here is an example script

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | from geometry\_msgs.msg import Twist    var\_twist = Twist()  pub\_position = rospy.Publisher('/cmd\_vel', Twist, queue\_size=1)  var\_twist.linear.x = 1  var\_twist.linear.y = 1  var\_twist.linear.z = 2  pub\_position.publish(var\_twist) |