Task 2.1: Problem Statement

blogpost-style , task2

aayushi #1 October 27, 2021, 4:52pm

Task 2.1: Waypoint Navigation in Auto Mission Mode

Overview

In this task, we will be learning the following:

- Getting started with px4 firmware
- Learning about various modes
- Sending Waypoints to px4 via mavros

In the previous task, we learned how to subscribe to a ros topic and publish the result on another rostopic. Now, we will be using this experience to communicate with px4 firmware via Mavros.

Prerequisites

Installations

- **PX4 and mavros installation**
  - If you have successfully installed PX4 and mavros in Task 0 in your catkin_ws, then you can continue with this task, but if you are facing any errors then make sure you rectify them by either going through the setup video and the documentation provided or by solving it with the help of QnA forum.

> Edit Nov '21: Current versions of PX4 require an additional command to function properly.
> After cloning the PX4 repo and before building the workspace, cd into it
>
> ```
> cd ~/catkin_ws/src/PX4-Autopilot
> ```
>
> and then run

> ```
> DONT_RUN=1 make px4_sitl_default gazebo
> ```
>
> This command enables PX4 to configure itself for working with Gazebo. The documentation file has been appended as well.

- **Install *QGroundControl***
  - On the command prompt enter:

```
sudo usermod -a -G dialout $USER
sudo apt-get remove modemmanager -y
sudo apt install gstreamer1.0-plugins-bad gstreamer1.0-libav gstreamer1.
```

  - Logout and login again to enable the change to user permissions.
  - Download QGroundControl.AppImage.
  - Install (and run) using the terminal commands:

```
chmod +x ./QGroundControl.AppImage
./QGroundControl.AppImage  (or double click)
```

- Update the strawberry_stacker package from github, follow the process to do so

```
cd ~/catkin_ws/src/strawberry_stacker
git add .
git commit -m "Put any message here"
git pull origin master
```

- Build your workspace

```
cd ~/catkin_ws
catkin build
```

Learning resources

- By this time, it is assumed you have completed the task 1.
- We highly recommend you to go through px4 official documentation (following till Modules & Commands will be enough for this task) and for mavros go through mavros px4 documentation before proceeding with the actual task as the learning curve might be steap in the start.
- This task will require the knowledge of *custom modes*, *custom messages* and *mavros integration*
- A walkthrough video is shared at the bottom of this page for tips and tricks do watch the video for getting familiar with px4 and MAVROS

Problem statement

- You need to make the drone in Gazebo to follow 4 waypoints in the ***mission*** mode of px4

- Create a *rosnode* named `waypoint_Mission` in a python script, which will set the waypoints to be sent to px4

- You need to call the *rosservice* `/mavros/cmd/arming` to arm and `/mavros/set_mode` to set mode of the drone to ***mission*** mode

- Then you have to call the *rosservice* `/mavros/mission/push` and `/mavros/mission/pull` to Request parameter from device (or internal cache) and send parameters from ROS to FCU respectively.

- The waypoints are as follows

  - Takeoff at the home position to 10 meters
  - Go to 19.134641, 72.911706, 10
  - Go to 19.134617, 72.911886, 10
  - Go to 19.134434, 72.911817, 10
  - Go to 19.134423, 72.911763, 10
  - Land at the last coordinate

Procedure

- Complete the boiler plate script named *waypoint_mission.py* provided to you in the *scripts* folder

- Launch the Gazebo world by typing the following command in a terminal

  ```
  roslaunch task_2 task2_1.launch
  ```

  Once the simulation window launches, you should see a drone in the gazebo environment.

- Run your python script in a separate terminal to start sending the waypoints and navigate the drone.

Expected Output

- As soon as you start the launch file, your python scripts should start running and the drone should arm, changes its mode to ***mission*** and then fly to the given coordinates and then land at the last coordinate.

Recording and Submission

- ROS allows us to record a log of the messages that occurred in a given time period. This is like recording a data stream. The ROS utility which does this is called **rosbag** , and the command to capture the data is `rosbag record`

- Before recording the rosbag, make sure you have completed the task and you are ready with the expected output.

- You can either run your python scripts manually or you can add the rosnode in the *task2_1.launch* file by adding the following lines before the `<group>` tag

  ```
  <node name="waypoint_mission" type="waypoint_mission.py" pkg="task_2" /
  ```

- You can run the `rosbag record` command separately on the command line. But to not loose any data you will have to start recording precisely at the same moment you start publishing messages. Hence it is a much more preferable option to include the rosbag recording in the launch file itself. It has already been added in the launch file and you need to enable it using the arg `record:="true"`

- To record your submission, write the following command in new window as soon as you run your python script

```
roslaunch task_2 task2_1.launch record:="true" rec_name:="waypoint_missi
```

- This will record and generate a bag file named *waypoint_mission.bag*
- Make sure the recording is complete, to verify that the recording was done, look for this message in the terminal after 5 minutes ie 300 seconds

```
[rosbag_record_waypoint_misson-6] process has finished cleanly
```

- After the recording is done, you will find the bag file in the folder named *bag_files* in the package
- Navigate to the folder where the bag file is saved and verify it.

```
rosbag info <NameOfBagFile>.bag
```

Submission instructions

- Rename all your python scripts with a prefix <SS_team_id>, example **SS_1234_send_waypoints.py**.
- Also rename the bag file to <SS_team_id>.bag, eg. **SS_1234.bag**
- Overall, your directory should look like this

```
__SS_1234.zip
|__SS_1234_send_waypoints.py
|__SS_1234.bag
```

- Submit the zip file on portal
- Create a video of your screen executing the task, upload the video on youtube unlisted with name **<SS_team_id>_task2.1**. And Submit the link on portal.

Walkthrough Video

All the best !

6 Likes

[Task 2 : Getting started with px4]

[SS: Task 2.1 launched!]

[Roslaunch error in Task 2.2]

[Error while lauching task_2]

[Task 2 live interaction session]

Smit unlisted #2 November 10, 2021, 8:59am

Smit closed #3 November 10, 2021, 9:00am