

Task 5 : Problem Statement

blogpost-style, task-5

aayushi  e-Yantra Staff

Jan 31

Task 5: Theme Implementation

Overview

In the previous task, we learned how to operate two drones simultaneously to pick and drop boxes. In this task, instead of having static boxes, you will see boxes appearing in a particular fashion. You have to scan and detect their location, pick them up and deliver to the desired truck in 5 mins.

[Skip to main content](#)

Prerequisites

Installations

- 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 all the previous tasks.
 - To get an overview about offboard mode and setpoint navigation you can go through the [px4 official documentation](#) .
 - We recommend you to go through the resources for Aruco detection provided in [Task 1.1](#) .
 - To get familiar with the Multi-Vehical system of PX4 you can go through the [Official PX4 documentation](#) .
 - In this task you can see a complete Strawberry farm setup, with 15 rows of strawberry plants numbered from 1-15, strawberry boxes of 2 colors ie. red and blue, 2 trucks (one blue and one red) on the side of the farm and 2 drones named **edrone0** and **edrone1** on their respective starting points.
 - The two different kinds of strawberry boxes have different significance. The premium quality strawberries are packed in blue colored boxes and normal quality strawberries are kept in red colored boxes. The aim here is to deliver red colored strawberry boxes on red truck and blue colored boxes to blue truck.
 - The farm consists of rows of strawberry plants followed by empty row spaces
- [Skip to main content](#) berry boxes will be kept as and when they are packed. This is

the same area where the eDrone has to come and land on the boxes to pick them up. These empty rows are spaced 4m apart, and the location of the 1st row is the same as the starting position of edrone0 and the location of the last row is same as the starting location of **edrone1**. Since the distance between each row and the location of the 1st row is known to you, you can calculate the coordinate of any row.

- The delivery trucks have a grid drawn on them where each cell is the desired location for the strawberry box to be placed. The exact location of 1st cell is given to you and the length and breadth of each cell is also given to you. So you can calculate the exact coordinates of any given cell.

“ Note: The delivery strategy is upto you whether you want to stack boxes on top of each other or you want to spread your boxes evenly. Note that for now there are less boxes in the scene but the number boxes will increase in the future tasks, so you might need to stack them as per the situation.

- After you have starting coordinates of each row, you will see boxes being spawned in a pseudo random manner in the gazebo environment. There are two types of boxes, blue and red, with aruco marker of different aruco id.
- You have to determine the exact position of the box (you are provided only the row number, use the strategy used in task 3.2 to locate the box in a row) pick it and deliver it to one of the trucks.
- Setpoints that are required:
 - Home position of **edrone0** : -1m 1m
 - Home position of **edrone1** : -1m 61m
 - Space between 2 rows : 4m
 - 1st cell of blue truck : 13.85m -7.4m 1.7m
 - 1st cell of red truck : 56.5m 64.75m 1.7m
 - Length of cells on truck : 0.85m
 - Width of cells on truck : 1.23m
 - Run time: 5 mins (300 sec)

“ Note: All the coordinates are given in the global reference, so if you want the eDrone0 to go to red truck, subtract the initial coordinates from

[Skip to main content](#) n to get your setpoint for the eDrone0. eg. setpoint for

eDrone0 to reach blue truck = $[13.85 - (-1) \text{ m}, -7.4 - 1 \text{ m}, 1.7 - 0 \text{ m}] = [14.85, -8.4\text{m}, 1.7\text{m}]$. Similarly you can calculate setpoint for any position since you have starting position of eDrones in global reference.

You can add your own setpoints to make the navigation of drones smooth.

- Maintain minimum height of the eDrone's at least 3m from ground while cruising (the higher you go, more context is captured in the camera but at the same time more time is taken to travel, so optimize your cruising height).
- Strategy to deploy two drones to complete the task is completely up to your creativity, you can use:
 - One drone to scan and send locations of box and second for pick and place
 - Scanning and pick and place from both the drones simultaneously
 - Or any other method that might help you to deliver maximum number of boxes in 5 mins.

Procedure

- Write a python script named *strawberry_stacker.py* in the scripts folder and create a rosnode named *strawberry_stacker* in this python script.
- Launch the Gazebo world by typing the following command in a terminal

```
roslaunch task_5 task5.launch
```

Once the simulation window launches, you should see complete farm setup in gazebo environment.

- Since there are multiple eDrones in the scene now, the names of all the *rostopics* have changed. Each *rostopic* has a prefix as the eDrone number. For eg. `/mavros/setpoint_position/local` becomes `/edrone0/mavros/setpoint_position/local` for edrone0 and `/edrone1/mavros/setpoint_position/local` for edrone1. This applies to all other *rostopics* including `/gripper/check`, `/camera/image_raw` etc. This also applies to *rosservices*. For exact list of *rostopics* and *rosservices* you can run the command `rostopic list` and `rosservice list` after

[Skip to main content](#) e.

- You can access the row numbers where the boxes are spawned by subscribing to the topic `/spawn_info`, message type is **UInt8**
- For picking up the boxes with eDrones, you need to call the appropriate *rosservice* for each eDrone and similarly for dropping the boxes.
- Run your python script in a separate terminal to start the task.

Expected Output

As soon as you start the launch file, your python script should start running . The eDrones should start picking and delivering the boxes to their respective trucks. Everything should be done in offboard mode. eDrones should come back to initial position and disarm.

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 **rosvbag** , and the command to capture the data is `rosvbag record`
- Before recording the rosvbag, 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 `rosvnode` in the *task5.launch* file by adding the following lines before the `<group>` tag

```
<node name="strawberry_stacker" type="strawberry_stacker
```

- You can run the `rosvbag 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 rosvbag 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

[Skip to main content](#)

```
roslaunch task_5 task5.launch record:="true" rec_name:="s
```



- This will record and generate a bag file named *strawberry_stacker.bag*

```
[roslaunch task_5 task5.launch record:="true" rec_name:="s
```



- 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.

```
roslaunch task_5 task5.launch record:="true" rec_name:="s
```

Submission instructions

- Rename all your python scripts with a prefix `<SS_team_id>`, example **SS_1234_strawberry_stacker.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_strawberry_stacker.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>_task5` and submit the link on portal.

All the best !

[🔗 Unclear Problem Statement](#)

[🔗 SS: Task 5 Launched!](#)

[Skip to main content](#)

UNLISTED ON FEB 2

dhnanj Participant

Feb 2

Do we have to pick and place all 30 boxes within 5min or as many boxes as possible within 5min?

[@aayushi](#) [@Smit](#) [@rishi](#)

CLOSED ON FEB 3



**Task 5: Theme
Implementation**

All the best !