




# Task 3.2 : Problem Statement

blogpost-style, task-3

aayushi  e-Yantra Staff

18d

 Table of contents

## Task 3.2: Pick n Place using ArUco detection

### Overview

In this task, we will be learning the following:

- Picking and placing in gazebo environment using drone.
- Setpoint Navigation in offboard mode.
- Interfacing of ArUco detection and setpoint navigation

In the previous task, we learned how to pick an place an object in gazebo environment when the setpoints and the location of the object are given. In this task, we will use our prior knowledge of ArUco marker detection to get the position of the object and the perform pick n place.

### 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 the task 1.1 and 2.2.
- 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](#) .

[Skip to main content](#)

## Problem Statement

- The gazebo world consist of a drone and a strwaberry box with ArUco marker on it.
- You have to put the drone in Offboard mode and publish the positions of the drone as setpoints.
- The drone needs to do the following
  - Takeoff at the initial position to the height of 3m
  - Start moving towards the final setpoint 9m 0m 3m
  - While travelling scan for the strawberry box with ArUco marker. As soon as the drone reaches above the box, land on it and pick up the box.
  - Again takeoff to the height of 3m and continue to move towards 9m 0m 3m
  - Land at 9m 0m 0m and drop the box
  - Takeoff again to height of 3m
  - Head back to the start position ie 0m 0m 3m
  - Finally land the drone at 0m 0m 0m and then disarm
- After landing on the box, you need to check that the drone is above the box in the allowable range to pick the box. To do that, you need to subscribe to the rostopic `/grripper_check`. If the value is *true*, you can now pick the box and if the value is *false*, the drone is not correctly placed and you need to correct the position.
- If the result of gripper\_check is *true*, to pick the box, you need to call the rosservice `/activate_gripper` and pass the value *true* to attach the box. If the box is attached, you will get a result from the rosservice as *true*. If the result is *false*, which means the drone is not correctly placed over the box.
- To detach the box, you need to use the same rosservice `/activate_gripper` and pass the value *false*. This will detach the box from the drone.

## Procedure

- Write a python script named `pick_n_place_aruco.py` in the scripts folder and create a rosnod named `pick_n_place_aruco` in this python script.
- Launch the Gazebo world by typing the following command in a terminal

```
roslaunch task_3 task3_2.launch
```

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

- Run your python script in a separate terminal to start sending the setpoints 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 **offboard** and then fly to the given coordinates, detect arUco marker with the help of ros camera stream on topic `/eDrone/camera/image_raw`, attach the box and fly to the said position and deliver it there. Drone should come back to the 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 rosnod in the `task3_2.launch` file by adding the following lines before the `<group>` tag

```
<node name="pick_n_place_aruco" type="pick_n_place_aruco.py" pkg="task_3" />
```

- 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

```
roslaunch task_3 task3_2.launch record:="true" rec_name:="pick_n_place_aruco.bag"
```

- This will record and generate a bag file named `pick_n_place_aruco.bag`
- Make sure the recording is complete, to verify that the recording was done, look for this message in the terminal after 3 minutes

```
[roslaunch record_pick_n_place_aruco] 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.

```
roslaunch record_pick_n_place_aruco <NameOfBagFile>.bag
```

#### Submission instructions

- Rename all your python scripts with a prefix <SS\_team\_id>, example **SS\_1234\_pick\_n\_place\_aruco.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_pick_n_place_aruco.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>\_task3\_2 and submit the link on portal.

**All the best !**

---

#### [🔗 SS Task 3: Pick\\_n\\_Place](#)



##### Task 3.2: Pick n Place using ArUco detection

Overview

Prerequisites

**Installations**

Learning resources

Problem Statement

Procedure

Expected Output

Recording and Submission

Submission instructions

**All the best !**

---

UNLISTED ON DEC 27, '21

---

CLOSED ON DEC 27, '21