**Program to take off the drone and land in new location:**

from dronekit import connect, VehicleMode, LocationGlobalRelative

import time

# Connect to the vehicle

vehicle = connect('udp:127.0.0.1:14550')

# Arm and take off

vehicle.mode = VehicleMode("GUIDED")

vehicle.armed = True

vehicle.simple\_takeoff(10)

# Wait for the drone to reach a certain altitude

while True:

altitude = vehicle.location.global\_relative\_frame.alt

if altitude >= 9.5: # target altitude - 0.5 meters

break

time.sleep(1)

# Move the drone to a new location

new\_location = LocationGlobalRelative(37.793105, -122.398768, 20)

vehicle.simple\_goto(new\_location)

# Wait for the drone to reach the new location

while True:

distance = vehicle.location.global\_relative\_frame.distance\_to(new\_location)

if distance <= 1: # target radius in meters

break

time.sleep(1)

# Land the drone

vehicle.mode = VehicleMode("LAND")

# Close the connection

vehicle.close()

**Program to simulate a mission using series of waypoints**

from dronekit import connect, VehicleMode, LocationGlobalRelative

import time

# Connect to the vehicle

vehicle = connect('udp:127.0.0.1:14550')

# Arm and take off

vehicle.mode = VehicleMode("GUIDED")

vehicle.armed = True

vehicle.simple\_takeoff(10)

# Wait for the drone to reach a certain altitude

while True:

altitude = vehicle.location.global\_relative\_frame.alt

if altitude >= 9.5: # target altitude - 0.5 meters

break

time.sleep(1)

# Define the mission waypoints

waypoints = [

LocationGlobalRelative(37.793105, -122.398768, 20),

LocationGlobalRelative(37.793109, -122.398824, 20),

LocationGlobalRelative(37.793095, -122.398857, 20),

LocationGlobalRelative(37.793057, -122.398843, 20),

LocationGlobalRelative(37.793042, -122.398797, 20),

LocationGlobalRelative(37.793050, -122.398751, 20),

LocationGlobalRelative(37.793084, -122.398722, 20),

LocationGlobalRelative(37.793119, -122.398724, 20)

]

# Fly the mission

for wp in waypoints:

vehicle.simple\_goto(wp)

while True:

distance = vehicle.location.global\_relative\_frame.distance\_to(wp)

if distance <= 1: # target radius in meters

break

time.sleep(1)

# Land the drone

vehicle.mode = VehicleMode("LAND")

# Close the connection

vehicle.close()

**Program to test the control algorithm using PID algorithm:**

from dronekit import connect, VehicleMode, LocationGlobalRelative

import time

# Connect to the vehicle

vehicle = connect('udp:127.0.0.1:14550')

# Arm and take off

vehicle.mode = VehicleMode("GUIDED")

vehicle.armed = True

vehicle.simple\_takeoff(10)

# Wait for the drone to reach a certain altitude

while True:

altitude = vehicle.location.global\_relative\_frame.alt

if altitude >= 9.5: # target altitude - 0.5 meters

break

time.sleep(1)

# Define the PID controller

class PIDController:

def \_\_init\_\_(self, kp, ki, kd, setpoint):

self.kp = kp

self.ki = ki

self.kd = kd

self.setpoint = setpoint

self.error = 0

self.error\_integral = 0

self.error\_derivative = 0

self.last\_error = 0

self.last\_time = time.time()

def update(self, measured\_value):

current\_time = time.time()

elapsed\_time = current\_time - self.last\_time

self.error = self.setpoint - measured\_value

self.error\_integral += self.error \* elapsed\_time

self.error\_derivative = (self.error - self.last\_error) / elapsed\_time

output = self.kp \* self.error + self.ki \* self.error\_integral + self.kd \* self.error\_derivative

self.last\_error = self.error

self.last\_time = current\_time

return output

# Define the control algorithm

def control\_algorithm(wp):

pid = PIDController(0.1, 0.05, 0.01, wp.alt)

while True:

altitude = vehicle.location.global\_relative\_frame.alt

output = pid.update(altitude)

vehicle.simple\_goto(LocationGlobalRelative(wp.lat, wp.lon, output))

time.sleep(1)

if abs(altitude - wp.alt) <= 0.5: # target altitude - 0.5 meters

break

# Test PID control

waypoints = [

LocationGlobalRelative(37.793105, -122.398768, 20),

LocationGlobalRelative(37.793109, -122.398824, 30),

LocationGlobalRelative(37.793095, -122.398857, 25),

LocationGlobalRelative(37.793057, -122.398843, 35),

LocationGlobalRelative(37.793042, -122.398797, 30),

LocationGlobalRelative(37.793050, -122.398751, 25),

LocationGlobalRelative(37.793084, -122.398722, 35),

LocationGlobalRelative(37.793119, -122.398724, 30)

]

for wp in waypoints:

control\_algorithm(wp)

# Land the drone

vehicle.mode = VehicleMode("LAND")

# Close the connection

vehicle.close()

This program implements a PID controller to control the altitude of the drone while flying a mission consisting of a series of waypoints. The **PIDController** class computes the control signal based on the difference between the setpoint and the measured value, as well as the integral and derivative of the error over time. The **control\_algorithm** function uses the PID controller to adjust the altitude of the drone to the target altitude of the current waypoint, while keeping the drone's position constant. You can modify this program to tune the PID gains, test different waypoint configurations, or evaluate the performance of the controller using appropriate metrics, such as tracking