Task-3

Name-Avinash khalkho

Roll-25/A14/014

Objective:

Develop a python script for Autonomous navigation of UAV to reach a set of checkpoints by controlling their velocity during the flight. You need to use the concept of PID Tuning to decrease or increase the speed of UAV around each checkpoint. The minimum safe checkpoint reach radius is 5m. You must bring the velocity of UAV within safe limits i.e:>10m/s and the maximum speed UAV can achieve is 80m/s. (Assume Flight Altitude)

Solution:

1. Learning about PID

From the YouTube playlist, I learned that a PID controller is a feedback control system. It calculates the error(difference between the current and desired value) and tries to minimize it using three components:

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(\tau)\,\mathrm{d}\tau + K_{\mathrm{d}} \frac{\mathrm{d}e(t)}{\mathrm{d}t},$$

This is the formula for a PID system

e(t) -> This functions returns the error at time t

- **Kp (Proportional):** Provides action proportional to distance between the current position and the desired output. For example, if Kp is high, the drone moves faster toward the checkpoint; if Kp is low, the drone may take a longer time to reach the checkpoint.
- **Ki (Integral):** Responds to accumulated past errors. For example, if the drone consistently slows down 0.5 m before reaching the target, Ki will accumulate these errors and slightly increase the drone’s speed so that it reaches the checkpoint.

- **Kd (Derivative):** Reacts to the rate of change of error. As the drone approaches the target, the error decreases rapidly, so Kd reduces the speed to prevent overshoot. For example, a high Kd value will slow down the drone near the target.

Approach

I read the article '**http://www.movabletype.co.uk/scripts/latlong.html**' and came to know that the drone is not going to fly in a Cartesian plain but on the real world and we are given latitude longitude and not coordinates. So after finding what haversine formula is I created a function **find_distance()** to find distance between two cords.

Then i made a function to move towards the target by updating the current coordinates.

Then by learning about the **pid equation** i coded it in python by creating a uav class and adding functions to it

Function description

- **__init__() >** Constructor to initialise the drone object of the **UAV class.** It initialises the kp,ki,kd values which will be used in the PID EQUATION; it is used to initialise,current position,current velocity,target,checkpoints,distanceleft to target and dt.

- **Find_distance() >** finds the distance between two latitude and longitudinal coordinates using haversine formula.

- **pid () >** computes the final velocity using PID equation.

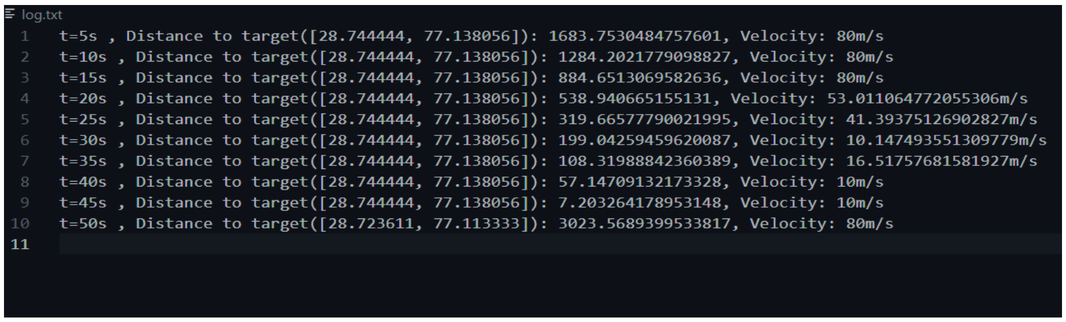
- **Move_real() >** Function to simulate movement of drone considering we have spherical coordinates. Updates its current position by calculating the distance to be travelled according to the velocity returned by the pid.

- **Run() >** Main function that updates the checkpoint which is reached logs the velocity every 5 second and simulates movement of drone,considering it is at constant altitude

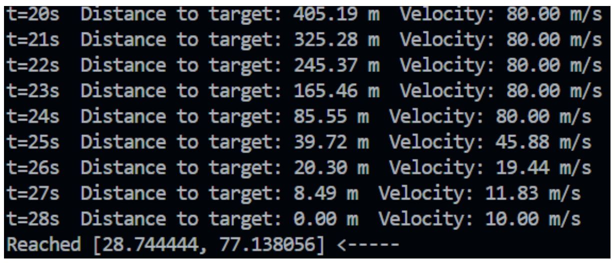
Variable description

Variable	Type	Desc
curr	list [lat, lon]	Current position of the UAV
c_v	float	Current velocity of the UAV (controlled by PID)
target	list [lat, lon]	Current checkpoint the UAV is moving toward
checkpoints	list [lat, lon]	Remaining checkpoints the UAV needs to reach
dleft	float	Distance remaining to the current target
kp, ki, kd	float	PID gains: proportional, integral, and derivative
integral	float	Accumulated error for integral term in PID
prev_err	float	Previous error for derivative calculation in PID
dt	float	Time step for updating UAV movement
t	float	Simulation time counter

Demo



```
t=5s , Distance to target([28.744444, 77.138056]): 1683.7530484757601, Velocity: 80m/s
t=10s , Distance to target([28.744444, 77.138056]): 1284.2021779098827, Velocity: 80m/s
t=15s , Distance to target([28.744444, 77.138056]): 884.6513069582636, Velocity: 80m/s
t=20s , Distance to target([28.744444, 77.138056]): 538.940665155131, Velocity: 53.011064772055306m/s
t=25s , Distance to target([28.744444, 77.138056]): 319.66577790021995, Velocity: 41.39375126902827m/s
t=30s , Distance to target([28.744444, 77.138056]): 199.04259459620087, Velocity: 10.147493551309779m/s
t=35s , Distance to target([28.744444, 77.138056]): 108.31988842360389, Velocity: 16.51757681581927m/s
t=40s , Distance to target([28.744444, 77.138056]): 57.14709132173328, Velocity: 10m/s
t=45s , Distance to target([28.744444, 77.138056]): 7.203264178953148, Velocity: 10m/s
t=50s , Distance to target([28.723611, 77.113333]): 3023.5689399533817, Velocity: 80m/s
```



```
t=20s  Distance to target: 405.19 m  Velocity: 80.00 m/s
t=21s  Distance to target: 325.28 m  Velocity: 80.00 m/s
t=22s  Distance to target: 245.37 m  Velocity: 80.00 m/s
t=23s  Distance to target: 165.46 m  Velocity: 80.00 m/s
t=24s  Distance to target: 85.55 m  Velocity: 80.00 m/s
t=25s  Distance to target: 39.72 m  Velocity: 45.88 m/s
t=26s  Distance to target: 20.30 m  Velocity: 19.44 m/s
t=27s  Distance to target: 8.49 m  Velocity: 11.83 m/s
t=28s  Distance to target: 0.00 m  Velocity: 10.00 m/s
Reached [28.744444, 77.138056] <-----
```