NAME - VISHAL KUMAR MAHATHA

REG NO - 20BRS1168

COURSE - Drone applications and Assembly

LAB - 5

1) CODE:

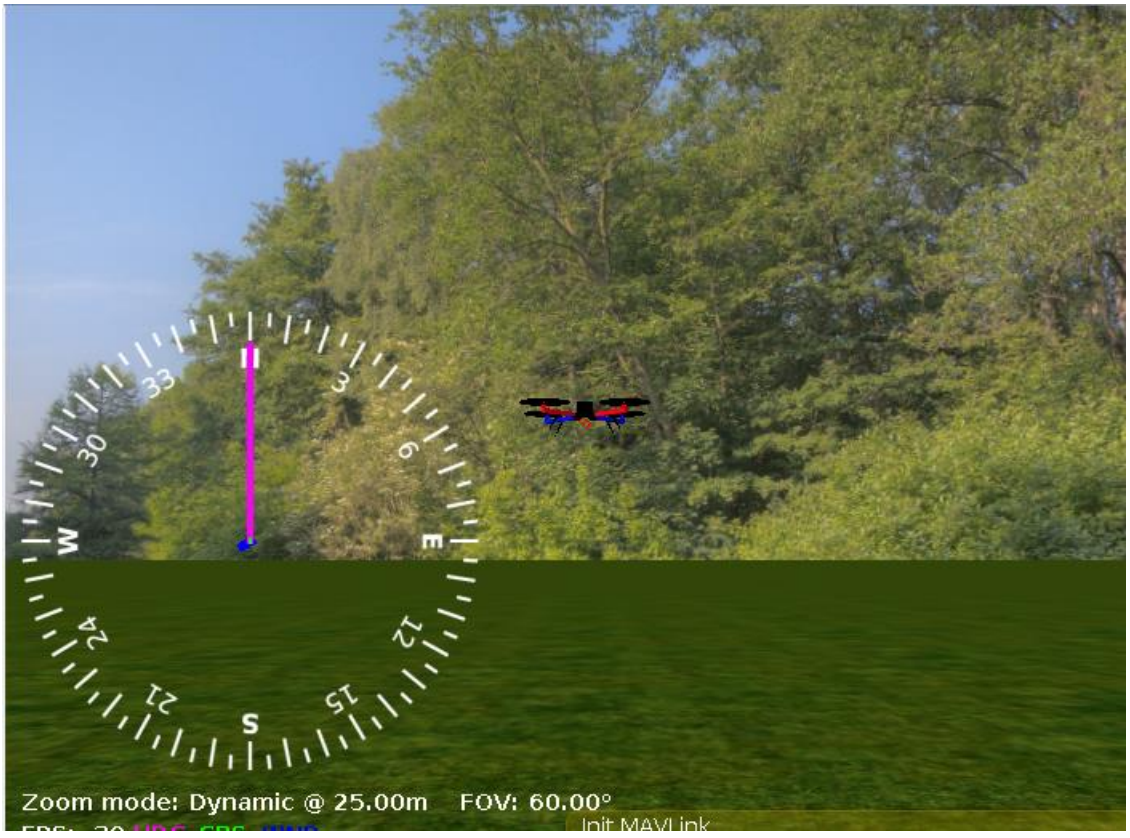
```
dronekit import connect, VehicleMode, LocationGlobalRelative
import time

# 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:
```

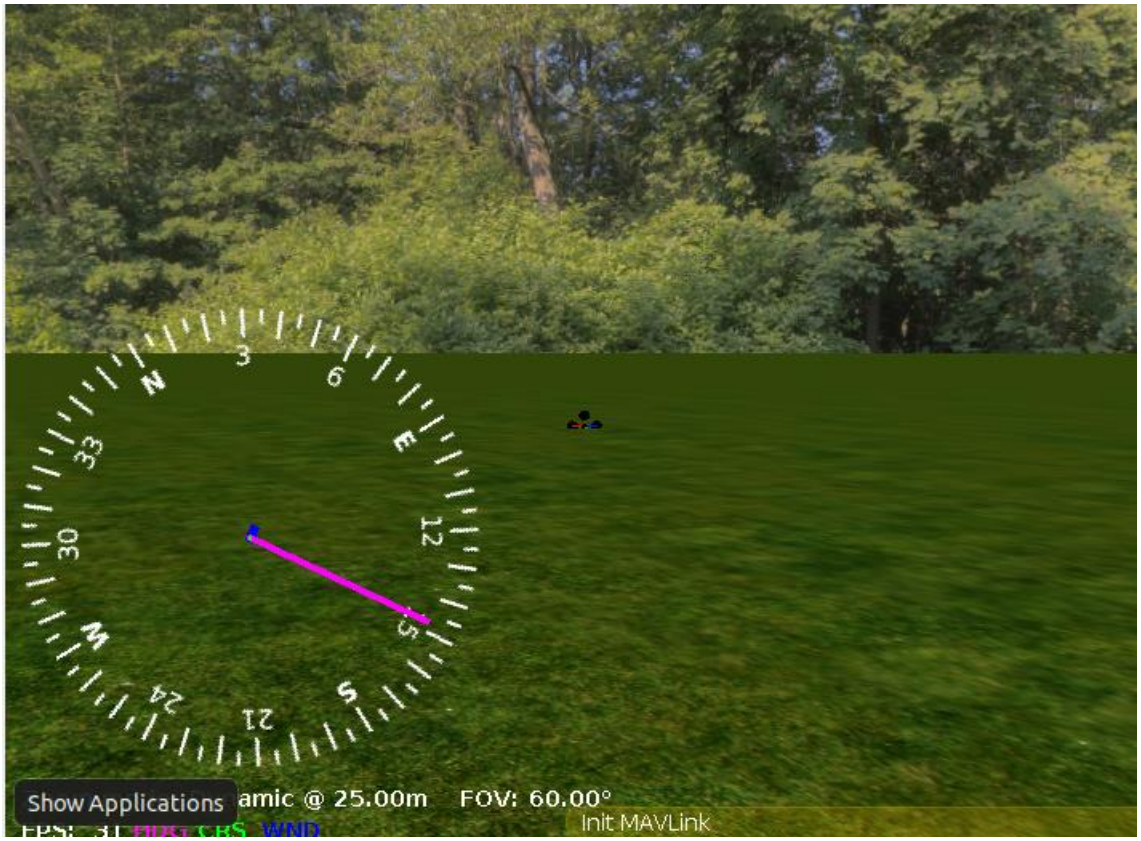
```
    distance = vehicle.location.global_relative_frame.distance_to(new_location)
    if distance <= 1:  # target radius in meters
        break
    time.sleep(1)
# Land the drone
vehicle.mode = VehicleMode("LAND")
# Close the connection
vehicle.close()
```

OUTPUT:

```
vishal@vishal-Vi:~$ python3 dr_1.py
Unknown mode 'GUIDED'
```

```
INFO  [commander] Ready for takeoff!
INFO  [commander] Armed by external command
INFO  [tone_alarm] arming warning
INFO  [commander] Disarmed by auto preflight disarming
INFO  [tone_alarm] notify neutral
INFO  [logger] closed logfile, bytes written: 6275964
```





2)CODE:

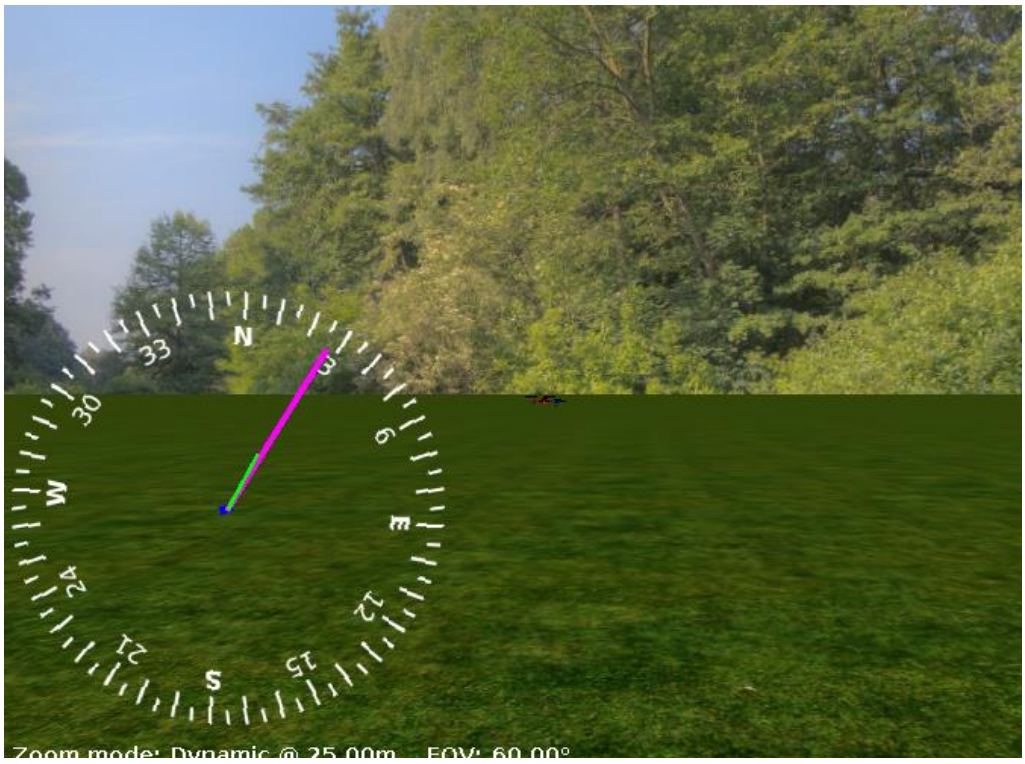
```
from dronekit import connect, VehicleMode, LocationGlobalRelative

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()
```

OUTPUT:

```
vishal@vishal-Vi:~$ python3 dr_2.py
Unknown mode 'GUIDED'
CRITICAL:autopilot:Failsafe activated
C                ot:Compass needs calibration - Land now!
INFO  [mavlink] partner IP: 127.0.0.1
INFO  [tone_alarm] notify negative
INFO  [tone_alarm] home set
WARN  [health_and_arming_checks] Preflight: GPS fix too low
INFO  [commander] Ready for takeoff!
```

Show Applications



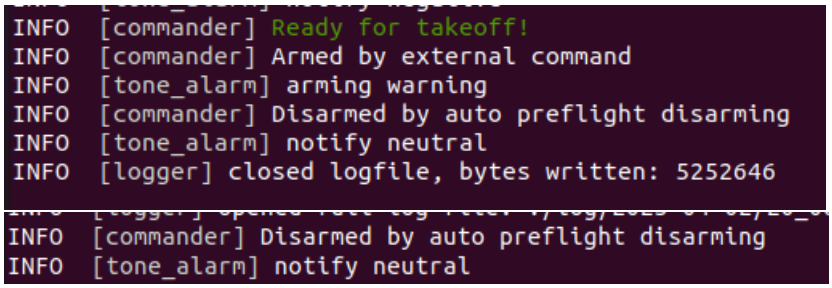
3) **CODE:**

```
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")
```

OUTPUT:




33
3
30
6
W
E
24
12
21
15
S
Show Applications
amic @ 25.00m
FOV: 60.00°