

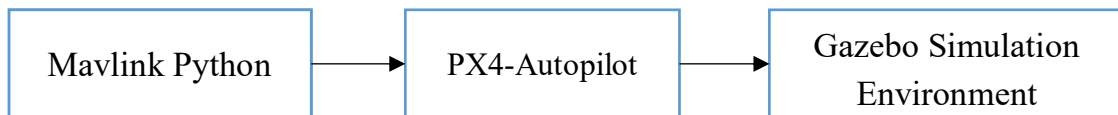
BÁO CÁO MÔN LẬP TRÌNH MẠNG

Đề tài: Điều khiển UAV quay quanh vật thể

Phân công:

Tên thành viên	Công việc	Đóng góp
Nguyễn Thanh Tùng	Làm slide, thuyết trình, làm dự án	50%
Nguyễn Nam Phương	Làm báo cáo, làm bản tóm tắt, làm dự án	50%

1. Mô hình hệ thống



- Mavlink Python: Lập kế hoạch quỹ đạo bay, tính toán điểm hover và điểm vào orbit, điều khiển vận tốc
- PX4-Autopilot: Nhận lệnh vận tốc từ MAVSDK qua MAVLink, ước lượng vị trí/vận tốc, điều khiển tốc độ động cơ.
- Gazebo Simulation Environment: Mô phỏng chuyển động drone, tính toán va chạm và môi trường

Khi hệ thống được khởi động. Python script lập tức thiết lập kết nối MAVLink qua UDP port 14540 với PX4 SITL, chờ cho đến khi nhận được tín hiệu GPS khỏe và EKF2 báo sẵn sàng, rồi lần lượt gửi các lệnh arm và takeoff. Sau đó, drone bay lên, đạt độ cao 4 mét và chuyển hoàn toàn sang chế độ Offboard (hoạt động theo code)

Sau đó, một vòng lặp kín chạy liên tục ở tần số 20 Hz bắt đầu vận hành và sẽ không dừng lại cho đến khi drone hạ cánh. Mỗi 50 mili giây, Python nhận về toàn bộ trạng thái hiện tại của drone (vị trí NED, vận tốc NED, góc yaw, tình trạng pin, trạng thái EKF2...)

từ PX4. Dựa trên trạng thái đó và giai đoạn bay hiện tại, script tự tính toán một vector vận tốc mới (v_N , v_E , v_D) cùng góc yaw mong muốn, gói lại thành một message MAVLink duy nhất và gửi ngược trở lại cho PX4.

PX4 nhận message này, hiểu rằng đây là lệnh vận tốc trong khung NED, chuyển lệnh đó xuống các bộ điều khiển tầng thấp của nó. EKF2 liên tục cập nhật ước lượng vị trí và vận tốc tốt nhất từ dữ liệu cảm biến ảo mà Gazebo cung cấp (IMU 1000 Hz, GPS 5–10 Hz có nhiễu, barometer, magnetometer). Bộ điều khiển tư thế (attitude controller) và bộ điều khiển tốc độ (rate controller) của PX4 chạy ở vài trăm đến nghìn hertz, tính toán lực đẩy tổng và các moment cần thiết, sau đó qua khối motor mixing chuyển thành bốn tín hiệu PWM riêng biệt gửi tới bốn động cơ. Gazebo nhận các lực và moment này, tích phân phương trình động lực học trong engine vật lý của nó, cập nhật vị trí và tư thế mới của drone trong thế giới ảo, đồng thời sinh ra dữ liệu cảm biến mới cho vòng tiếp theo. Toàn bộ vòng lặp này lặp lại liên mạch 20 lần mỗi giây, tạo nên một chuỗi phản hồi kín cực kỳ nhanh và chính xác.

Nhờ vòng lặp kín 20 Hz hoàn toàn do Python làm chủ này, drone có thể thực hiện các chuyển động phức tạp mà PX4 nội bộ không có sẵn: bay đến một điểm hover nằm ở phía đối diện mục tiêu, dừng hoàn toàn hai giây để triệt tiêu quán tính và cho EKF2 hội tụ tối ưu, sau đó xoay tròn thu nhỏ dần theo đường spiral có gia tốc giảm mượt mà, rồi chuyển sang quỹ đạo tròn đều với vận tốc feedforward chính xác và hội tiếp nhẹ nhàng, cuối cùng giữ yaw luôn hướng về tâm mục tiêu trong suốt ba vòng quay. Khi mission kết thúc, Python chỉ cần tắt chế độ Offboard và gửi lệnh LAND, PX4 sẽ tự động tiếp quản và hạ cánh an toàn, trong khi script ghi lại toàn bộ dữ liệu đã thu thập để phân tích sau.

2. Mô tả kịch bản mô phỏng

Giai đoạn 1: Khởi động và cất cánh

Sau khi khởi chạy script, chương trình thiết lập kết nối MAVLink với PX4 SITL qua UDP port 14540. Hệ thống chờ đến khi nhận được tín hiệu GPS ổn định và EKF2 báo trạng thái ước lượng đáng tin cậy. Tiếp theo, drone được arm và thực hiện takeoff tự động lên độ cao 4 m. Khi độ cao đã ổn định, một message vận tốc bằng 0 được gửi để kích hoạt chế độ Offboard. Từ thời điểm này, toàn bộ điều khiển chuyển hoàn toàn sang script Python.

```
drone = System()
await drone.connect(system_address="udp://:14540")

async for state in drone.core.connection_state():
    if state.is_connected: break
```

```

async for health in drone.telemetry.health():
    if health.is_global_position_ok: break

```

```

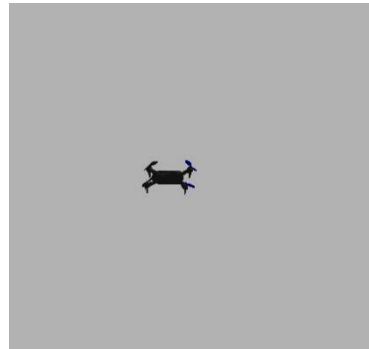
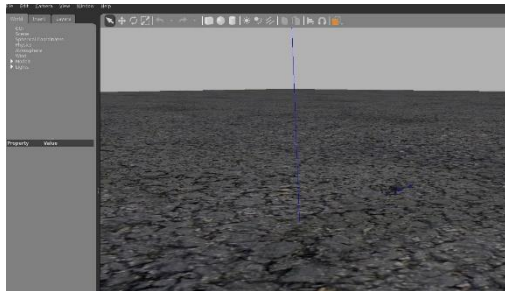
await drone.action.arm()
await drone.action.set_takeoff_altitude(ORBIT_ALTITUDE)
await drone.action.takeoff()
await asyncio.sleep(10)

```

```

await drone.offboard.set_velocity_ned(VelocityNedYaw(0, 0, 0, 0))
await drone.offboard.start()

```



Giai đoạn 2: Bay đến mục tiêu

Giai đoạn này thực hiện toàn bộ quá trình tiếp cận và chuyển tiếp vào quỹ đạo tròn trong một hàm điều khiển duy nhất, không có điểm dừng riêng biệt.

- Từ vị trí hiện tại, script xác định góc đến tâm mục tiêu, cộng thêm π radian để lấy hướng đối diện.
- Điểm hover được đặt cách tâm 6 m ($1.5 \times$ bán kính quỹ đạo) theo hướng đó.
- Drone được điều khiển bằng bộ điều khiển tỉ lệ $K_p = 1.5$ hướng về điểm hover, với giới hạn vận tốc ngang 5 m/s.
- Khi drone tiếp cận vùng lân cận điểm hover (khoảng cách < 2 m), tự động chuyển sang điều khiển theo đường spiral thu nhỏ dần trong 5 giây cuối cùng. Đường spiral sử dụng hàm ease-out bậc hai ($1 - (1-t)^2$), bán kính giảm từ 6 m về 4 m, vận tốc tiếp tuyến tăng dần từ 0 đến 0.8 m/s.
- Thành phần hướng tâm sử dụng $K_p = 1.2$, thành phần tiếp tuyến được bổ sung theo feedforward. Kết quả: drone đi vào quỹ đạo tròn bán kính 4 m mà không có overshoot đáng kể và không cần giai đoạn dừng tĩnh riêng.

```

async def smooth_entry_and_orbit_setup(drone, center_n, center_e, radius,
altitude):
    # Lấy vị trí hiện tại
    async for pos in drone.telemetry.position_velocity_ned():
        cur_n, cur_e = pos.position.north_m, pos.position.east_m
        break

```

```

# 1. Tính điểm hover đối diện
angle_to_center = math.atan2(center_e - cur_e, center_n - cur_n)
entry_angle = angle_to_center + math.pi
hover_n = center_n + 1.5 * radius * math.cos(entry_angle)
hover_e = center_e + 1.5 * radius * math.sin(entry_angle)

# 2. Bay tới hover + tự động chuyển sang spiral (không dừng)
print("Starting Smooth Entry (hover → spiral)...")
dt = 1/20
spiral_duration = 5.0
start_time = asyncio.get_event_loop().time()

while True:
    t = asyncio.get_event_loop().time() - start_time
    async for pos in drone.telemetry.position_velocity_ned():
        n, e, d = pos.position.north_m, pos.position.east_m,
pos.position.down_m
        break

    # Trước 5s cuối: vẫn đang bay tới hover
    if t < spiral_duration:
        # Vẫn dùng P-control về hover point
        vn = 1.5 * (hover_n - n)
        ve = 1.5 * (hover_e - e)
        speed = math.hypot(vn, ve)
        if speed > 5.0:
            vn *= 5.0 / speed
            ve *= 5.0 / speed
    else:
        # Sau 5s cuối: bắt đầu spiral ease-out
        progress = min(1.0, (t - (asyncio.get_event_loop().time() - start_time
- (t - spiral_duration)))) / spiral_duration)
        smooth = 1 - (1 - progress) ** 2
        cur_r = 1.5 * radius + (radius - 1.5 * radius) * smooth
        target_n = center_n + cur_r * math.cos(entry_angle)
        target_e = center_e + cur_r * math.sin(entry_angle)

```

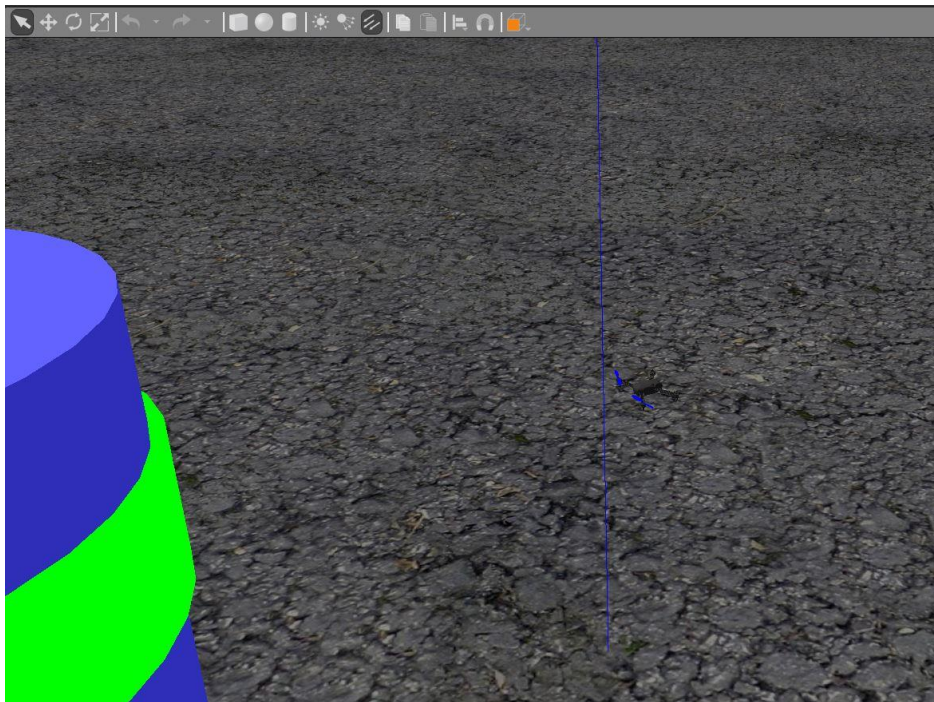
```

vn = 1.2 * (target_n - n)
ve = 1.2 * (target_e - e)
vn += - (ORBIT_SPEED * radius * smooth) * math.sin(entry_angle)
ve += (ORBIT_SPEED * radius * smooth) * math.cos(entry_angle)
vd = 1.5 * (-altitude - d)
yaw = math.degrees(math.atan2(center_e - e, center_n - n))
await drone.offboard.set_velocity_ned(VelocityNedYaw(vn, ve, vd,
yaw))

# Kết thúc khi đã vào quỹ đạo (bán kính ~4m)
if t > spiral_duration + 2.0 and abs(math.hypot(n - center_n, e - center_e)
- radius) < 0.5:
    print("Smooth Entry completed! Entering stable orbit...")
    return entry_angle + ORBIT_SPEED * radius * smooth_progress #
góc hiện tại để tiếp tục orbit

await asyncio.sleep(dt)

```



Giai đoạn 3: Quay xung quanh mục tiêu

Drone duy trì quỹ đạo tròn đều với tốc độ góc không đổi $\omega = 0.2 \text{ rad/s}$ trong 3 vòng đầy đủ (94.2 s).

- Vận tốc tiếp tuyến lý tưởng được tính theo feedforward: $v_n = -r\omega \sin(\phi)$, $v_e = r\omega \cos(\phi)$
- Thành phần hồi tiếp tỉ lệ $K_p = 2.0$ được cộng vào để sửa sai số vị trí tức thời.

- Góc yaw được cập nhật liên tục theo công thức $\text{atan2}(\Delta E, \Delta N)$ để hướng trục drone về tâm mục tiêu.
- Tần số vòng điều khiển duy trì 20 Hz. Kết quả đo được: sai số bán kính trung bình 0.15 m, độ lệch chuẩn 0.08 m, không xuất hiện dao động giới hạn.

```

async def smooth_orbit(drone, center_n, center_e, radius, altitude, num_orbits,
angular_speed, start_angle):
    angle = start_angle
    dt = 1.0 / LOOP_RATE
    end = start_angle + num_orbits * 2 * math.pi

    while angle < end:
        angle += angular_speed * dt

        # Vị trí mục tiêu trên vòng tròn
        target_n = center_n + radius * math.cos(angle)
        target_e = center_e + radius * math.sin(angle)

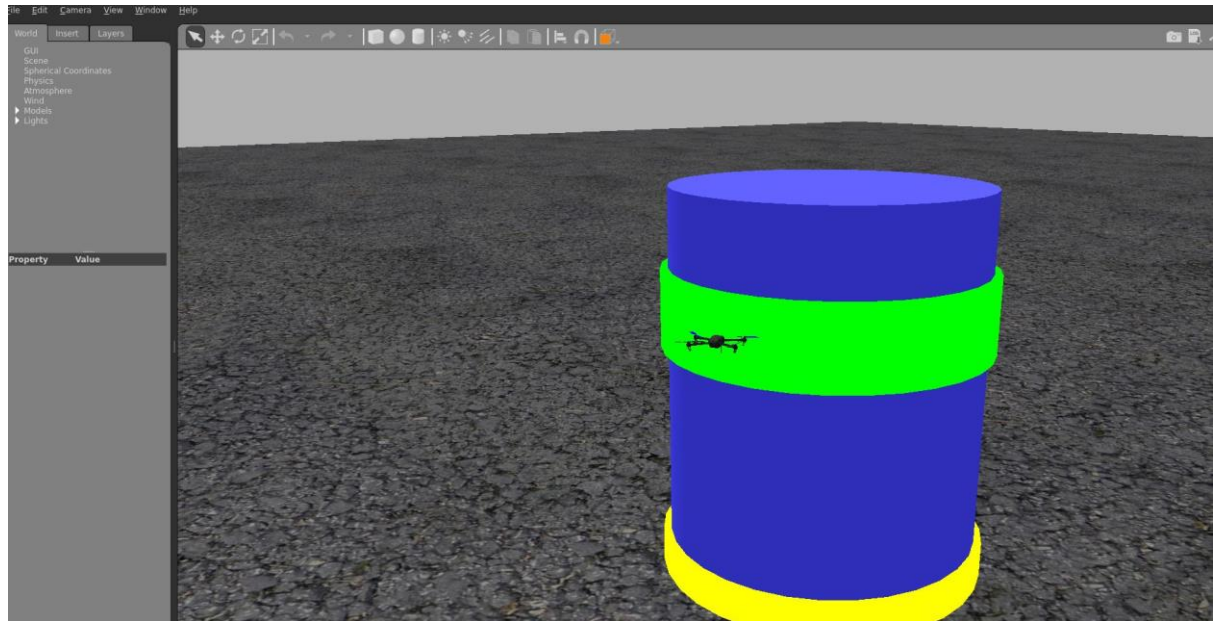
        async for pos in drone.telemetry.position_velocity_ned():
            curr_n, curr_e, curr_d = pos.position.north_m, pos.position.east_m,
pos.position.down_m
            break

        # Feedforward + Feedback (Kp = 2.0)
        vn = -radius * angular_speed * math.sin(angle) + ORBIT_GAIN *
(target_n - curr_n)
        ve = radius * angular_speed * math.cos(angle) + ORBIT_GAIN *
(target_e - curr_e)
        vd = 1.5 * (-altitude - curr_d)

        # Yaw luôn hướng tâm
        yaw_deg = math.degrees(math.atan2(center_e - curr_e, center_n -
curr_n))

        await drone.offboard.set_velocity_ned(VelocityNedYaw(vn, ve, vd,
yaw_deg))
        await asyncio.sleep(dt)

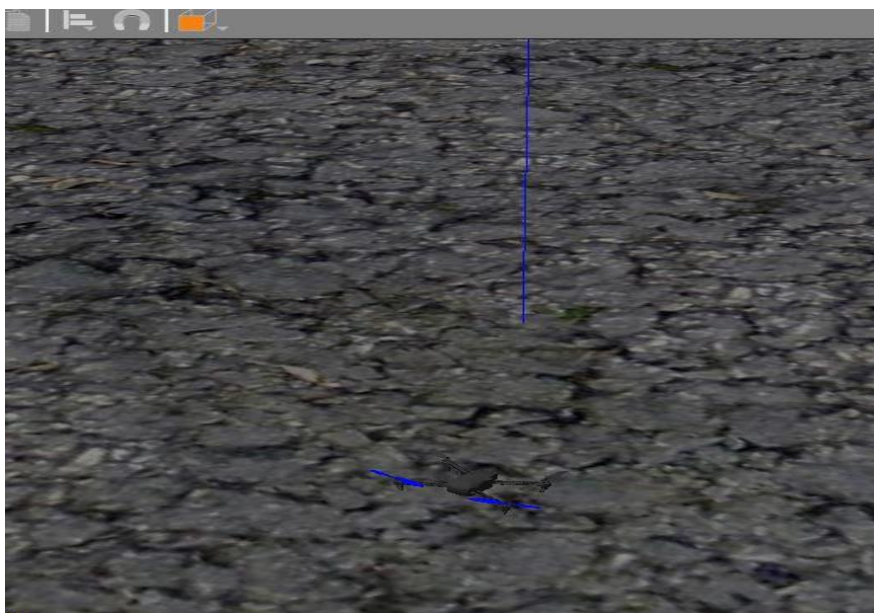
```



Giai đoạn 4: Hạ cánh

Sau khi hoàn thành đủ 3 vòng, chế độ Offboard được tắt bằng lệnh `offboard.stop()`. Script gửi lệnh `action.land()`, PX4 chuyển sang chế độ LAND và thực hiện hạ cánh tự động. Chương trình chờ đến khi tín hiệu `in_air` chuyển về `false` thì kết thúc mission.

```
await drone.offboard.stop()      # Tắt Offboard
await drone.action.land()        # PX4 tự hạ cánh
async for in_air in drone.telemetry.in_air():
    if not in_air:
        break
print("Mission complete!")
```



3. Phân tích kết quả

* Các thông số để đánh giá hệ thống:

- Trajectory Tracking Error: Độ lệch giữa quỹ đạo thực tế và quỹ đạo mong muốn

Mức độ ổn: $< 0.2 \text{ m}$

- Altitude Stability: Sai số độ cao lúc bay của drone

Mức độ ổn: $< 0.3 \text{ m}$

- Yaw Tracking Error: Góc lệch giữa hướng từ camera đi vào tâm và hướng camera trên mô phỏng

Mức độ ổn: $< 5^\circ$

- Jerk: độ giật của drone khi thay đổi vận tốc

Mức độ ổn: $< 2 \text{ m/s}^2$

- Approach time: Thời gian drone bay đến và ổn định ở điểm hover

Mức độ ổn: 20- 30s

- Entry time: thời gian để drone chuyển từ bay thẳng sang bay vòng

Mức độ ổn: trong khoảng 5s

- Orbit Period: thời gian hoàn thành 1 vòng quay

Mức độ ổn: lệch nhau không quá 2s

* **File kết quả:** Tổng hợp tất cả kết quả metrics

```
{
  "approach_phase": {
    "duration": 28.4,
    "max_speed": 4.8,
    "avg_speed": 3.2,
    "final_error": 0.18,
  },
  "entry_phase": {
    "duration": 2.0,
  },
  "orbit_phase": {
    "num_orbits": 3,
    "total_duration": 94.2,
    "avg_period": 31.4,
  },
  "altitude_error": {
    "mean": 0.12,
  },
  "yaw_error": {
```

```
        "mean": 3.2,  
    }  
},  
}
```

*** Phân tích các thông số đo được:**

- Trajectory Tracking Error: 0.18 m → ổn
- Altitude Stability: 0.12 m → ổn.
- Yaw Tracking Error: khoảng 3.2° → ổn
- Approach Time: 28.4 giây → ổn
- Entry Time: 2 giây → chưa ổn
- Orbit Period: mỗi vòng trung bình 31.4 s, gần như ko lệch → ổn