+

**TRƯỜNG ĐẠI HỌC BÁCH KHOA- ĐẠI HỌC ĐÀ NẴNG**

**🙡🙡 KHOA ĐIỆN TỬ- VIỄN THÔNG 🙣🙣**

A blue and yellow logo

Description automatically generated A logo of a company

AI-generated content may be incorrect.

**LẬP TRÌNH MẠNG**

**ĐỀ TÀI:**

**Xây dựng hệ thống điều khiển drone qua Web và truyền dữ liệu thời gian thực dựa trên MAVSDK và PX4 SITL**

GVHD : TS Nguyễn Văn Hiếu

SVTH : Lê Thành Duy - 106220250 - 22KTMT2

Nguyễn Ngọc Duy - 106220251 - 22KTMT2

**Đà Nẵng, tháng 11 năm 2025**

Muc luc

[1. Giới thiệu: 4](#_Toc215006904)

[2. Cơ sở lý thuyết: 4](#_Toc215006905)

[2.1 Hệ thống điều khiển bay PX4 4](#_Toc215006906)

[2.2 Giao thức MAVLink và MAVSDK 5](#_Toc215006907)

[2.3 Điều khiển Offboard và hệ tọa độ NED 5](#_Toc215006908)

[2.4 Mô phỏng UAV bằng Gazebo 6](#_Toc215006909)

[2.5 Giao diện Web và kiến trúc 3 lớp 7](#_Toc215006910)

[2.6 Chuỗi dữ liệu điều khiển và phản hồi 7](#_Toc215006911)

[2.7 Lý thuyết quỹ đạo bay (Pattern Trajectory Generation) 8](#_Toc215006912)

[3. Thiết kế hệ thống 9](#_Toc215006913)

[3.1 Kiến trúc tổng quát: 9](#_Toc215006914)

[3.2 Khối web UI 9](#_Toc215006915)

[3.2.1 Cơ chế gửi lệnh từ Web đến Flask 10](#_Toc215006916)

[3.2.2 Cơ chế điều khiển bằng bàn phím (Keyboard Mode) 11](#_Toc215006917)

[3.2.3 Cơ chế nhận và hiển thị telemetry 11](#_Toc215006918)

[3.3 Khối Flask API 11](#_Toc215006919)

[3.4 Khối MAVSDK Control 12](#_Toc215006920)

[3.5 Khối Simulator ( PX4 STIL + Gazebo ) 12](#_Toc215006921)

[3.6 Luồng xử lý của hệ thống 13](#_Toc215006922)

[4. Triển khai hệ thống 14](#_Toc215006923)

[4.1. Môi trường và kiến trúc phần mềm 14](#_Toc215006924)

[4.2. Tổ chức mã nguồn 14](#_Toc215006925)

[4.3. Triển khai Backend (Flask & MAVSDK) 14](#_Toc215006926)

[4.3.1. Kỹ thuật xử lý đa luồng cho MAVSDK 14](#_Toc215006927)

[4.4. Triển khai thuật toán bay theo mẫu 15](#_Toc215006928)

[4.5. Triển khai Giao diện Web & Điều khiển Bàn phím 16](#_Toc215006929)

[4.5.1. Cơ chế điều khiển bằng bàn phím (Velocity Control) 16](#_Toc215006930)

[4.5.2. Hiển thị Telemetry và Log 16](#_Toc215006931)

[5. Mô phỏng và chạy thử: 17](#_Toc215006932)

[5.1 Chuẩn bị môi trường mô phỏng 17](#_Toc215006933)

[5.2 Mục tiêu chạy thử: 17](#_Toc215006934)

[6. Kết quả mô phỏng hệ thống: 18](#_Toc215006935)

[6.1 Các lệnh cơ bản: 18](#_Toc215006936)

[6.2 Chế độ classic mode ( bay theo mẫu hình ): 19](#_Toc215006937)

[6.3 Keyboard mode (Điều khiển bằng bàn phím) 23](#_Toc215006938)

[7. Đánh giá hệ thống 25](#_Toc215006939)

[7.1 Độ chính xác của hệ thống điều khiển 25](#_Toc215006940)

[7.2 Độ trễ (latency) 26](#_Toc215006941)

[7.3 Telemetry và khả năng truyền thông MAVLink 27](#_Toc215006942)

[7.4 Tính ổn định của Offboard Mode 27](#_Toc215006943)

[8. Kết luận: 27](#_Toc215006944)

|  |  |  |
| --- | --- | --- |
| Họ và tên | Công việc | Tỷ lệ |
| Lê Thành Duy | Xây dựng giao diện Web UI (điều khiển, telemetry, log).  Lập trình phần keyboard mode.  Xây dựng Flask API và kết nối MAVSDK – PX4 SITL.  Làm báo cáo, slide. | 50% |
| Nguyễn Ngọc Duy | Xây dựng giao diện Web UI (điều khiển, telemetry, log).  Tích hợp gửi lệnh từ Web → Flask.  Lập trình phần classic mode( bay theo mẫu )  Làm báo cáo slide | 50% |

# **1. Giới thiệu:**

Trong những năm gần đây, các hệ thống mô phỏng điều khiển phương tiện bay không người lái (UAV) đã trở thành một công cụ quan trọng trong đào tạo và nghiên cứu về lập trình mạng. Việc điều khiển UAV thông qua các giao thức mạng giúp sinh viên tiếp cận trực tiếp với những vấn đề thực tiễn như truyền thông bất đồng bộ, độ trễ, an toàn giao tiếp và quản lý kênh điều khiển. Bên cạnh đó, mô phỏng bằng PX4 SITL và Gazebo cho phép kiểm thử các thuật toán điều khiển mà không cần sử dụng thiết bị phần cứng thực tế, giảm thiểu rủi ro và chi phí. Đề tài xây dựng một hệ thống điều khiển UAV theo mô hình pipeline bao gồm các thành phần: trình duyệt web chạy trên Windows, máy chủ Flask hoạt động trong môi trường WSL, thư viện MAVSDK để giao tiếp với PX4, bộ mô phỏng PX4 SITL và môi trường mô phỏng Gazebo. Người dùng thực hiện thao tác điều khiển UAV trực tiếp từ trình duyệt, bao gồm gửi lệnh cất cánh, hạ cánh, di chuyển và bay theo các mẫu quỹ đạo định sẵn. Các thông số trạng thái như độ cao, tốc độ và dung lượng pin được thu thập từ PX4 và hiển thị thời gian thực trên giao diện web.

Hệ thống được thiết kế theo hướng tách biệt giữa giao diện người dùng, máy chủ trung gian và bộ mô phỏng. Cách tiếp cận này giúp đề tài đáp ứng các mục tiêu: minh họa rõ ràng quy trình điều khiển UAV qua mạng, tạo môi trường thực hành an toàn, và cung cấp nền tảng để mở rộng sang các bài toán điều khiển nâng cao trong tương lai.

# **2. Cơ sở lý thuyết:**

## **2.1 Hệ thống điều khiển bay PX4**

PX4 là một nền tảng autopilot mã nguồn mở, được phát triển rộng rãi trong nghiên cứu và công nghiệp UAV. PX4 bao gồm hệ thống điều khiển vị trí, điều khiển attitude, hệ thống hợp nhất dữ liệu cảm biến (sensor fusion) thông qua EKF2/EKF3, và nhiều chế độ bay tự động như Stabilized, Position, Mission, Offboard.

Trong ngữ cảnh mô phỏng, PX4 hoạt động dưới dạng SITL (Software In The Loop). Thay vì chạy trên bo mạch vật lý như Pixhawk, PX4 SITL chạy trên máy tính, mô phỏng toàn bộ logic điều khiển bằng phần mềm. Điều này đảm bảo tất cả thuật toán điều khiển, logic trạng thái và chế độ bay đều giống với phiên bản trên drone thật. SITL giao tiếp với mô hình động lực học trong Gazebo. PX4 đọc dữ liệu mô phỏng từ cảm biến (IMU, GPS, barometer) và gửi lệnh điều khiển động cơ lại cho simulator. Chính quá trình lặp này tạo nên chuyển động chân thực của UAV trong không gian 3D.

## **2.2 Giao thức MAVLink và MAVSDK**

MAVLink (Micro Air Vehicle Link) là giao thức truyền thông dạng message-based được dùng trong tất cả UAV dựa trên PX4 và ArduPilot. MAVLink hoạt động qua UDP/TCP hoặc Serial và định nghĩa hàng trăm kiểu thông điệp như position, velocity, attitude, mode, heartbeat, command.

Trong hệ thống này:

* PX4 SITL phát MAVLink qua cổng UDP 14540
* MAVSDK lắng nghe và giao tiếp với PX4
* Flask không trực tiếp đọc MAVLink → Flask chỉ giao tiếp qua MAVSDK

MAVSDK Python

MAVSDK là thư viện bậc cao, giúp gửi lệnh MAVLink và đọc telemetry một cách dễ dàng, không cần xử lý byte MAVLink thủ công.

Một số API quan trọng:

1. action  
   Chứa các lệnh mức cao: arm, takeoff, land, return\_to\_launch.
2. offboard  
   Cho phép gửi lệnh vị trí/tốc độ liên tục (≥ 2 Hz) để duy trì Offboard.
3. telemetry  
   Cung cấp dữ liệu GPS, độ cao, attitude, battery, flight mode.

MAVSDK đảm nhiệm vai trò chuyển đổi lệnh từ Python sang MAVLink chuẩn, nhờ đó quá trình điều khiển drone trở nên an toàn và ổn định.

## **2.3 Điều khiển Offboard và hệ tọa độ NED**

Trong PX4, Offboard là chế độ điều khiển mà UAV nhận lệnh từ một máy tính bên ngoài thay vì từ RC hoặc chế độ bay tự động. Đây là chế độ lý tưởng cho việc lập trình quỹ đạo và điều khiển theo thời gian thực.

Yêu cầu của PX4 đối với Offboard:

* Drone phải được gửi lệnh liên tục ≥ 2 Hz
* Nếu ngừng gửi > 0.5 giây → PX4 tự động thoát khỏi Offboard
* Chỉ cho phép chuyển sang Offboard khi drone đang ở trạng thái arm

Hệ tọa độ NED (North – East – Down)

MAVSDK dùng NED để định nghĩa chuyển động:

* Trục X (North): hướng Bắc địa phương
* Trục Y (East): hướng Đông
* Trục Z (Down): trục thẳng đứng hướng xuống

Ví dụ:

Muốn drone bay lên 5 m → gửi Z = -5

Muốn drone bay 3 m sang phải → Y = 3

Tất cả quỹ đạo hình học (square, circle, star…) đều được xây dựng dựa trên hệ quy chiếu này.

## **2.4 Mô phỏng UAV bằng Gazebo**

Gazebo là engine mô phỏng vật lý mạnh mẽ được PX4 hỗ trợ mặc định. Nó mô phỏng:

* Lực nâng, lực cản khí động học
* Trọng lực và mô men quán tính
* Va chạm vật thể
* Môi trường 3D và cảm biến mô phỏng

Mô hình “Iris” hoặc “Typhoon H480” là hai mô hình PX4 cung cấp. Khi PX4 gửi tín hiệu điều khiển động cơ (PWM/forces), Gazebo tính toán và cập nhật vị trí drone trong không gian.

Kết quả mô phỏng từ Gazebo được phản ánh:

* hình ảnh 3D
* dữ liệu GPS mô phỏng
* độ cao từ barometer mô phỏng
* tốc độ từ IMU mô phỏng

Điều này giúp việc kiểm tra quỹ đạo và hành vi drone trở nên trực quan.

## **2.5 Giao diện Web và kiến trúc 3 lớp**

Giao diện Web là thành phần đầu tiên mà người dùng tương tác. Nó được viết bằng HTML/JavaScript và gửi lệnh qua REST API đến Flask.

Kiến trúc chia thành 3 lớp rõ ràng:

Frontend (index.html)

* Hiển thị nút điều khiển (Arm, Takeoff, Land, RTL…)
* Gửi lệnh bằng fetch() qua HTTP
* Hiển thị telemetry lấy từ Flask
* Cho phép điều khiển bằng bàn phím (WASD, QE, ZX)

Backend Flask (app.py)

* Tiếp nhận request từ UI
* Chạy event loop MAVSDK
* Truyền lệnh xuống PX4
* Quản lý trạng thái máy bay
* Gửi telemetry về Web UI

PX4 SITL + MAVSDK

* Xử lý thực thi lệnh
* Báo về telemetry
* Điều khiển drone trong mô phỏng

## **2.6 Chuỗi dữ liệu điều khiển và phản hồi**

Hiểu rõ chuỗi dữ liệu là điều bắt buộc khi lập trình hệ thống điều khiển UAV.

Lệnh điều khiển (Control Path)

User → Browser → HTTP → Flask → MAVSDK → MAVLink → PX4 → Gazebo

Ví dụ: Takeoff

1. Người dùng nhấn “Takeoff”
2. UI gửi POST /takeoff
3. Flask gọi drone.action.takeoff()
4. MAVSDK gửi MAVLink COMMAND
5. PX4 chuyển mode TAKEOFF
6. Drone tăng độ cao trong Gazebo

Luồng telemetry (Feedback Path)

PX4 → MAVLink → MAVSDK → Flask → UI

UI sẽ cập nhật mỗi 1 giây các thông số như:

* Flight mode
* GPS (lat, lon)
* Relative altitude
* Battery percent

## **2.7 Lý thuyết quỹ đạo bay (Pattern Trajectory Generation)**

Các mẫu bay được xây dựng bằng công thức toán học trong không gian NED.

Ví dụ:

Hình tròn:

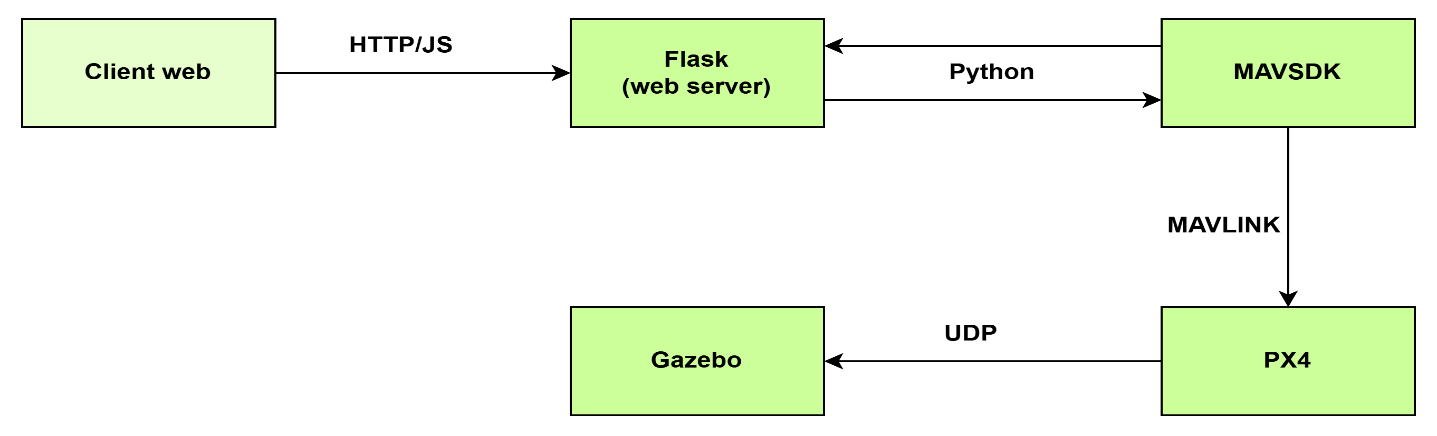
Hình số 8 (lemniscate):

Hình trái tim (cardioid):

Drone được dẫn theo từng điểm với tần số 10–20 Hz, tạo ra chuyển động mượt mà trong mô phỏng.

# **3. Thiết kế hệ thống**

## **3.1 Kiến trúc tổng quát:**



Hệ thống được thiết kế theo mô hình phân tầng, trong đó mỗi khối đảm nhiệm một chức năng riêng biệt nhưng phối hợp chặt chẽ để đảm bảo quá trình điều khiển UAV diễn ra ổn định và theo thời gian thực. Các khối chính bao gồm: khối giao diện Web UI, khối Flask API, khối MAVSDK Control và khối mô phỏng PX4 SITL kết hợp Gazebo. Mỗi khối có chức năng, luồng xử lý và cơ chế giao tiếp riêng như trình bày bên dưới.

## **3.2 Khối web UI**

Khối Web UI đóng vai trò là lớp giao tiếp trực tiếp giữa người dùng và hệ thống điều khiển UAV. Đây là điểm bắt đầu của toàn bộ pipeline Web → Flask → MAVSDK → PX4 SITL → Gazebo, nơi mọi thao tác điều khiển và phản hồi trạng thái được thực hiện. Giao diện Web được xây dựng dưới dạng một trang HTML tĩnh (index.html), kết hợp với CSS và JavaScript để tạo ra trải nghiệm điều khiển trực quan, dễ sử dụng và hoạt động ổn định trên mọi trình duyệt hiện đại.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

**Hình 1. Giao diện web UI**

Web UI bao gồm ba nhóm thành phần chính:

1. Nhóm điều khiển cơ bản (Basic Control Panel)  
   Cung cấp các nút lệnh như Arm, Disarm, Takeoff, Land, RTL, Emergency Stop.  
   Mỗi nút tương ứng với một sự kiện JavaScript và được gửi đến Flask thông qua phương thức HTTP POST.
2. Nhóm điều khiển nâng cao
   * Classic Mode: cho phép chọn các mẫu hình bay như hình vuông, tròn, trái tim, số 8, ngôi sao…
   * Keyboard Mode: điều khiển drone bằng bàn phím với các phím WASD + QE + ZX và cập nhật vận tốc theo thời gian thực.
3. Bảng hiển thị trạng thái (Telemetry Panel)  
   Hiển thị toàn bộ dữ liệu mà PX4 gửi về, bao gồm:
   * Flight mode
   * Tọa độ GPS mô phỏng
   * Độ cao tương đối
   * Vận tốc
   * Yaw, pitch, roll
   * Pin mô phỏng
   * Trạng thái arm/disarm

Các thông số này được cập nhật liên tục thông qua cơ chế polling (AJAX GET /telemetry).

### **3.2.1 Cơ chế gửi lệnh từ Web đến Flask**

Mỗi khi người dùng nhấn một nút điều khiển, Web UI sử dụng lệnh:

fetch('/takeoff', { method: 'POST' })

Tất cả lệnh từ Web UI đều được thiết kế dạng REST API, giúp việc truyền lệnh rõ ràng, tách biệt và dễ debug.  
Sau khi Flask tiếp nhận yêu cầu, hệ thống sẽ chuyển lệnh sang MAVSDK mà không cần bất kỳ thao tác mạng bổ sung nào.

### **3.2.2 Cơ chế điều khiển bằng bàn phím (Keyboard Mode)**

Để mô phỏng việc điều khiển UAV thời gian thực, Web UI sử dụng JavaScript để bắt sự kiện nhấn và thả phím:

document.addEventListener('keydown', handleKeydown);

document.addEventListener('keyup', handleKeyup);

Mỗi lần nhấn phím, UI gửi một gói điều khiển vận tốc theo dạng:

fetch('/velocity', {

method: 'POST',

headers: {'Content-Type': 'application/json'},

body: JSON.stringify({ vx, vy, vz, yaw\_rate })

});

Lệnh được gửi liên tục với tần số ~10–20 Hz để đảm bảo PX4 duy trì Offboard Mode ổn định. Đây là một kỹ thuật cần thiết trong điều khiển UAV thực tế, do PX4 sẽ tự động thoát Offboard nếu không nhận được lệnh điều khiển đều đặn.

### **3.2.3 Cơ chế nhận và hiển thị telemetry**

Web UI định kỳ gửi yêu cầu:

fetch('/telemetry')

và cập nhật toàn bộ bảng thông số theo thời gian thực. Điều này giúp người điều khiển nhìn thấy ngay độ cao, vận tốc, hướng bay, trạng thái pin và mode bay của UAV mà không cần mở QGroundControl.

Telemetry trên Web UI đồng bộ hoàn toàn với dữ liệu nhận được từ MAVSDK, giúp việc quan sát và đánh giá UAV thuận tiện hơn.

## **3.3 Khối Flask API**

Khối Flask API đóng vai trò trung gian giữa giao diện Web UI và MAVSDK. Đây là nơi tiếp nhận các yêu cầu điều khiển từ phía client và chuyển chúng thành các lời gọi hàm trong MAVSDK. Flask cũng cung cấp các endpoint RESTful để phục vụ việc trao đổi dữ liệu.

Các API chính bao gồm:

* POST /arm, /takeoff, /land, /move: thực hiện các lệnh điều khiển UAV.
* GET /telemetry: trả về dữ liệu trạng thái của UAV như độ cao, tốc độ, battery và GPS.
* Các hàm xử lý: chuyển đổi request HTTP thành lời gọi hàm Python tương ứng trong MAVSDK.

Flask không giao tiếp với MAVSDK bằng một giao thức mạng nào, mà thực hiện thông qua lời gọi hàm nội bộ (in-process function calls). Điều này giúp giảm độ trễ và đảm bảo lệnh điều khiển được gửi đến PX4 một cách nhanh nhất.

## **3.4 Khối MAVSDK Control**

MAVSDK là tầng điều khiển cốt lõi của hệ thống. Đây là thư viện lập trình cấp cao được xây dựng trên giao thức MAVLink, cung cấp các hàm API giúp điều khiển UAV một cách thuận tiện mà không cần thao tác trực tiếp với cấu trúc gói MAVLink.

Chức năng chính của khối này bao gồm:

* Kết nối với PX4 SITL qua giao thức MAVLink (UDP tại cổng 14540).
* Thực hiện các lệnh điều khiển: arm, disarm, takeoff, land, set\_position, set\_velocity…
* Thu thập dữ liệu telemetry: attitude, altitude, vị trí, tốc độ, dung lượng pin và chế độ bay.
* Cung cấp hàm bất đồng bộ (async/await) cho phép xử lý song song và không chặn luồng Flask.

Trong luồng xử lý, khi Flask gọi hàm MAVSDK, MAVSDK sẽ gửi lệnh tương ứng đến PX4 SITL. Sau đó, MAVSDK cũng nhận telemetry do PX4 gửi về và cung cấp cho Flask khi Flask yêu cầu dữ liệu.

## **3.5 Khối Simulator ( PX4 STIL + Gazebo )**

Khối mô phỏng bao gồm hai thành phần then chốt: PX4 SITL và Gazebo.

* PX4 SITL đóng vai trò là flight controller mô phỏng, chạy đầy đủ các thuật toán điều khiển UAV như PID, EKF, attitude control, rate controller…
* Gazebo mô phỏng môi trường vật lý 3D, bao gồm trọng lực, va chạm, khí động học và cảm biến.

Luồng xử lý hoạt động như sau:

1. MAVSDK gửi lệnh điều khiển đến PX4 SITL qua MAVLink (UDP).
2. PX4 tiến hành tính toán điều khiển và gửi lệnh motor, mô phỏng cảm biến và trạng thái máy bay.
3. PX4 giao tiếp với Gazebo thông qua Ignition Transport hoặc UDP để mô phỏng các tín hiệu sensor như IMU, GPS, barometer.
4. Gazebo phản hồi lại cho PX4 các thông tin vật lý mô phỏng.
5. PX4 gửi telemetry trở lại MAVSDK → Flask → Web UI.

Khối mô phỏng đảm bảo mô tả chính xác hành vi của UAV trong thực tế mà không cần sử dụng thiết bị thật, giúp việc kiểm thử và phát triển hệ thống diễn ra an toàn và hiệu quả hơn.

## **3.6 Luồng xử lý của hệ thống**

Hệ thống điều khiển UAV được xây dựng trên mô hình pipeline gồm nhiều lớp liên kết với nhau theo thứ tự: Web UI → Flask API → MAVSDK → PX4 SITL → Gazebo. Mỗi lớp đảm nhận một vai trò riêng và trao đổi dữ liệu qua các giao thức hoặc cơ chế truyền thông phù hợp. Luồng xử lý tổng quát của hệ thống được mô tả như sau:

* Người dùng thao tác trên Web UI  
  Khi người dùng nhấn vào một nút điều khiển như *Arm*, *Takeoff*, *Land* hoặc các lệnh di chuyển, JavaScript trong Web UI tạo ra một yêu cầu HTTP POST gửi tới Flask API.
* Flask tiếp nhận và xử lý yêu cầu  
  Flask nhận request từ trình duyệt, phân tích tham số lệnh và gọi trực tiếp hàm tương ứng trong MAVSDK thông qua Python. Đây là quá trình xử lý nội bộ, không sử dụng giao thức mạng nên độ trễ nhỏ.
* MAVSDK gửi lệnh tới PX4 SITL  
  MAVSDK chuyển yêu cầu từ Flask thành một lệnh MAVLink và truyền tới PX4 SITL bằng giao thức UDP. PX4 tiếp nhận lệnh, thực hiện tính toán điều khiển và cập nhật trạng thái UAV mô phỏng.
* PX4 SITL cập nhật mô phỏng trên Gazebo  
  PX4 gửi các giá trị điều khiển (ví dụ: lực đẩy, moment, tín hiệu motor) đến Gazebo thông qua cơ chế Ignition Transport. Gazebo thực hiện mô phỏng bay và áp dụng các hiệu ứng vật lý lên UAV trong môi trường 3D.

# **4. Triển khai hệ thống**

Phần này trình bày chi tiết quá trình hiện thực hóa hệ thống điều khiển UAV "Advanced Drone Control Center". Hệ thống được xây dựng theo kiến trúc Client-Server, trong đó Frontend (Web) gửi lệnh điều khiển và Backend (Python Flask) xử lý logic giao tiếp với PX4 thông qua thư viện MAVSDK.

## **4.1. Môi trường và kiến trúc phần mềm**

Hệ thống được phát triển và vận hành trên các công cụ sau:

* Ngôn ngữ: Python 3.10 (Backend), JavaScript ES6 (Frontend).
* Framework: Flask (Web Server), MAVSDK-Python (Giao thức MAVLink).
* Mô phỏng: PX4 Autopilot (SITL) kết nối qua giao thức UDP.
* Xử lý bất đồng bộ: Sử dụng thư viện asyncio và threading để đảm bảo API phản hồi tức thời trong khi vẫn duy trì kết nối liên tục với UAV.

## **4.2. Tổ chức mã nguồn**

Dựa trên mã nguồn thực tế, cấu trúc dự án được tổ chức như sau:

* app.py: Trạm trung chuyển chính. Khởi tạo Flask Server, quản lý vòng lặp sự kiện (Event Loop) cho MAVSDK, và định nghĩa các API RESTful.
* patterns.py: Thư viện chứa thuật toán bay tự động. Định nghĩa các quỹ đạo phức tạp như: Hình vuông, Tam giác, Hình tròn, Vô cực (Infinity), Trái tim, và Xoắn ốc.
* index.html (Static Folder): Giao diện người dùng tích hợp "Glassmorphism UI", xử lý logic bàn phím và hiển thị Telemetry thời gian thực.

## **4.3. Triển khai Backend (Flask & MAVSDK)**

Điểm đặc biệt trong triển khai Backend là việc tích hợp cơ chế bất đồng bộ (asyncio) của MAVSDK vào môi trường đồng bộ của Flask.

### **4.3.1. Kỹ thuật xử lý đa luồng cho MAVSDK**

Do Flask hoạt động theo cơ chế chặn (blocking), trong khi MAVSDK yêu cầu async/await, hệ thống sử dụng một luồng riêng (threading.Thread) để chạy asyncio event loop. Điều này giúp Server có thể lắng nghe lệnh HTTP từ Web UI mà không làm gián đoạn kết nối với UAV.

**4.3.2. API Điều khiển và Telemetry**

Backend cung cấp các Endpoint chính:

* Quản lý trạng thái: /arm, /disarm, /takeoff, /land, /rtl.
* Điều khiển vận tốc (Offboard): /velocity - Nhận vector vận tốc (Vx, Vy, Vz, Yaw) từ bàn phím và gửi lệnh set\_velocity\_body tới UAV.
* Bay theo mẫu: /pattern - Nhận tham số (loại hình, kích thước, độ cao) và kích hoạt hàm tương ứng trong patterns.py.
* Telemetry: /telemetry - Sử dụng asyncio.gather để thu thập đồng thời vị trí, pin, GPS, và chế độ bay nhằm giảm độ trễ dữ liệu.

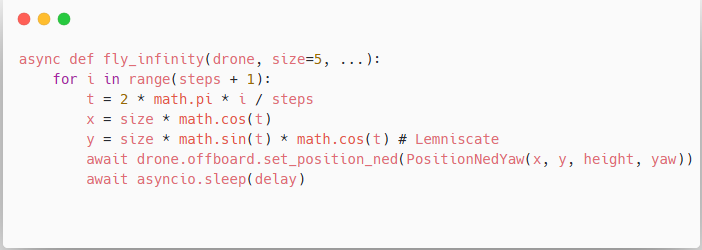
## **4.4. Triển khai thuật toán bay theo mẫu**

Module này sử dụng chế độ Offboard của PX4 để điều khiển vị trí chính xác (PositionNedYaw). Trước khi thực hiện quỹ đạo, hệ thống luôn gọi hàm prepare\_offboard để thiết lập điểm tham chiếu.

Các mẫu bay được triển khai bằng toán học lượng giác để tính toán tọa độ điểm đến (Setpoints):

* Hình tròn/Xoắn ốc:
* Hình trái tim
* Hình vô cực (Infinity): Dựa trên đường cong Lemniscate of Gerono.

\*Trích đoạn logic bay hình vô cực:



## **4.5. Triển khai Giao diện Web & Điều khiển Bàn phím**

Giao diện (index.html) được chia thành 2 chế độ: Classic Mode (Nút bấm) và Keyboard Mode (Điều khiển trực tiếp).

### **4.5.1. Cơ chế điều khiển bằng bàn phím (Velocity Control)**

Đây là tính năng nâng cao cho phép điều khiển UAV mượt mà như game mô phỏng. Quy trình xử lý như sau:

1. Lắng nghe sự kiện: JavaScript bắt sự kiện keydown và keyup để cập nhật trạng thái các phím (W, A, S, D, Q, E, Shift).
2. Tính toán Vector: Một vòng lặp setInterval (50ms/lần) liên tục kiểm tra trạng thái phím và cộng/trừ giá trị vào vector vận tốc:
   * W/S: Thay đổi vận tốc trục Y (Tiến/Lùi).
   * A/D: Thay đổi vận tốc trục X (Trái/Phải).
   * Z/X: Thay đổi vận tốc trục Z (Lên/Xuống).
   * Q/E: Thay đổi tốc độ xoay (Yaw Rate).
3. Gửi lệnh: Vector tổng hợp được gửi qua API /velocity. Backend nhận dữ liệu và truyền vào hàm set\_velocity\_body.

Sơ đồ luồng dữ liệu bàn phím:

Phím nhấn (Client) -> Update State (JS) -> JSON POST /velocity -> Flask -> MAVSDK VelocityBodyYawspeed -> PX4

### **4.5.2. Hiển thị Telemetry và Log**

Hệ thống sử dụng cơ chế Polling (gửi request định kỳ 1 giây/lần) để cập nhật thông số lên giao diện:

* Pin: Hiển thị thanh dung lượng với màu sắc thay đổi theo mức pin (Xanh > 50%, Vàng < 50%, Đỏ < 20%).
* Log hệ thống: Hiển thị danh sách các hành động (ARM, PATTERN START, ERROR) giúp người dùng dễ dàng debug.

# **5. Mô phỏng và chạy thử:**

Để kiểm chứng toàn bộ hệ thống điều khiển drone qua Web và đảm bảo các thành phần trong pipeline Web UI → Flask API → MAVSDK → PX4 SITL → Gazebo hoạt động chính xác, một kịch bản mô phỏng chi tiết đã được xây dựng và triển khai. Mục tiêu của kịch bản mô phỏng là tái hiện đầy đủ quá trình điều khiển UAV trong môi trường an toàn, đồng thời quan sát hành vi trong Gazebo và kiểm tra tính đúng đắn của dữ liệu telemetry trả về.

## **5.1 Chuẩn bị môi trường mô phỏng**

Môi trường mô phỏng được triển khai trên hệ điều hành Ubuntu (WSL2). PX4 SITL được cấu hình chạy với mô hình UAV mặc định Iris, sử dụng Gazebo làm engine mô phỏng động lực học. Khi khởi động, PX4 tạo cổng MAVLink tại địa chỉ udp://0.0.0.0:14540, cho phép MAVSDK Python kết nối trực tiếp.

Song song đó, Flask server được khởi chạy để cung cấp giao diện API cho Web UI. Web UI được truy cập thông qua trình duyệt, đảm bảo cách điều khiển giống như điều khiển drone thật qua một hệ thống điều khiển từ xa.

Trước khi chạy thử, các tệp mã nguồn chính (app.py, patterns.py, index.html) được tải vào đúng vị trí, Flask được khởi động bằng lệnh python3 app.py, còn PX4 SITL được chạy qua lệnh:

make px4\_sitl gz\_x500

Khi PX4 đã sẵn sàng, drone mô phỏng xuất hiện trong Gazebo chờ lệnh điều khiển.

## **5.2 Mục tiêu chạy thử:**

Kịch bản mô phỏng nhằm kiểm tra 4 nhóm chức năng cốt lõi:

1. Khả năng giao tiếp Web → Flask → MAVSDK → PX4:  
   Kiểm tra xem lệnh gửi từ Web UI có đến được PX4 và được PX4 thực thi đúng hay không.
2. Khả năng điều khiển cơ bản (cất cánh, hạ cánh, quay về):  
   Đảm bảo drone đáp ứng đúng các hành động của người dùng.
3. Khả năng duy trì Offboard Control:  
   Kiểm tra drone có giữ được Offboard khi gửi lệnh liên tục qua MAVSDK.
4. Khả năng bay theo quỹ đạo:  
   Đánh giá độ chính xác và độ ổn định của các mẫu bay như hình vuông, hình tròn, số 8, trái tim…

# **6. Kết quả mô phỏng hệ thống:**

## **6.1 Các lệnh cơ bản:**

Nhóm lệnh cơ bản bao gồm: Arm, Disarm, Takeoff, Land, Return-to-Launch (RTL) và Emergency Stop. Các lệnh này được gửi từ Web UI → Flask API → MAVSDK và được PX4 tiếp nhận dưới dạng MAVLink command.

Trong quá trình thử nghiệm, UAV mô phỏng phản hồi ổn định đối với tất cả các lệnh cơ bản:

* Arm: UAV chuyển từ trạng thái "disarmed" sang "armed" ngay lập tức, QGroundControl hiển thị đầy đủ thông tin trạng thái, và PX4 chuẩn bị các thành phần điều khiển bên trong (arming checks).
* Takeoff: Sau khi nhận lệnh, PX4 chuyển sang chế độ Takeoff Mode và UAV tăng độ cao ổn định đến mức mặc định (khoảng 2,5–3 m). Telemetry ghi nhận độ cao tăng đều, không vượt quá ngưỡng mục tiêu.
* Land: UAV giảm độ cao từ từ và tiếp đất mềm trong Gazebo. PX4 tự động chuyển sang chế độ Landing và giữ ổn định attitude cho đến khi tốc độ thẳng đứng tiệm cận về 0.
* Return-to-Launch (RTL): UAV quay về vị trí xuất phát và tự động hạ cánh. Quỹ đạo quay về được PX4 tính toán dựa trên GPS mô phỏng.
* Disarm: Sau khi hoàn thành quá trình hạ cánh, UAV được disarm an toàn.

Kết quả từ QGroundControl cũng cho thấy tất cả trạng thái được cập nhật chính xác, chế độ bay thay đổi đúng thứ tự (MANUAL → ARM → TAKEOFF → HOLD → LAND/RTL). Không xuất hiện lỗi về an toàn, lỗi arming check hoặc lỗi điều khiển trong suốt quá trình thử nghiệm.

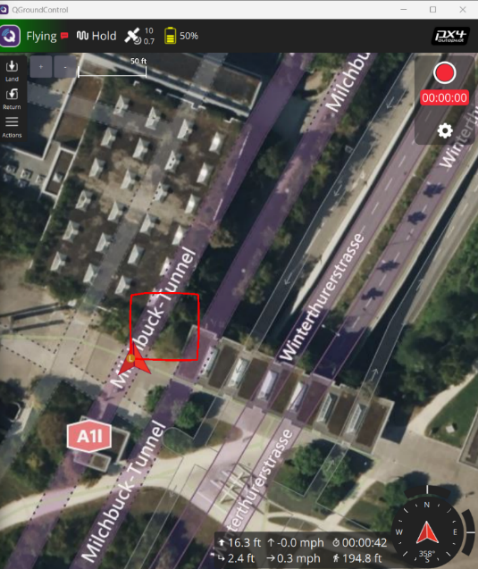
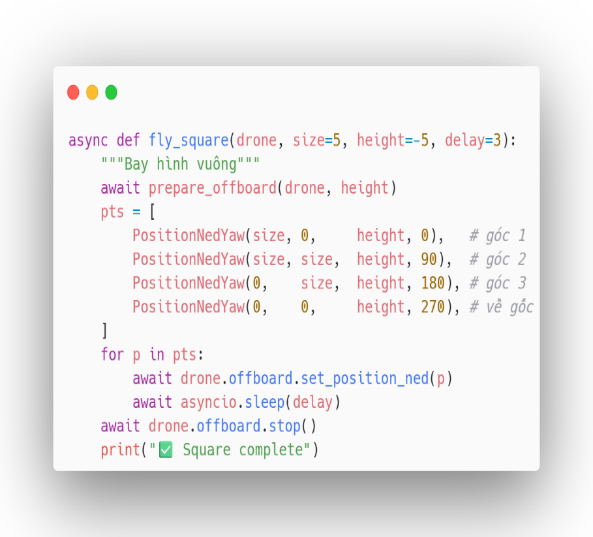
## **6.2 Chế độ classic mode ( bay theo mẫu hình ):**

Trong ứng dụng có nhiều mẫu để drone chạy theo mẫu như là: trái tim, hình vuông, ngôi sao,… Các hình được vẽ trong ứng dụng Qgroundcontrol, đây là ứng dụng nhận lệnh từ PX4 và MAVSDK để cất cánh, hạ cánh và bay theo hình mẫu dựa theo code và lệnh đã soạn sẵn.

A screenshot of a game

AI-generated content may be incorrect.

Trong quá trình thử nghiệm, mỗi mẫu hình được kích hoạt từ giao diện Web UI thông qua một lệnh POST gửi đến Flask API. Sau khi lệnh được truyền qua MAVSDK và được PX4 tiếp nhận, UAV mô phỏng bắt đầu bay theo quỹ đạo tương ứng. Tất cả chuyển động của UAV đều được quan sát đồng thời trên Gazebo (môi trường vật lý 3D) và QGroundControl (trình giám sát MAVLink). Các hình minh họa dưới đây là dữ liệu thu được từ QGroundControl. Mỗi quỹ đạo cho phép đánh giá những đặc điểm khác nhau của hệ thống:

* **Hình vuông :**

Quỹ đạo hình vuông cho phép kiểm tra khả năng điều khiển UAV theo các đoạn thẳng và góc rẽ 90 độ. Trong mô phỏng, UAV di chuyển qua bốn điểm theo đúng thứ tự, tốc độ ổn định và sai số vị trí rất nhỏ. Điều này chứng minh hệ thống xử lý tốt các lệnh định vị theo trục X–Y trong hệ tọa độ NED.

* **Hình tròn:**

****A screenshot of a map

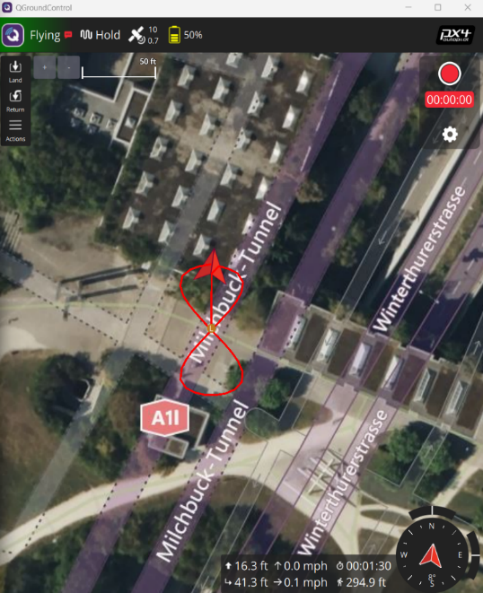
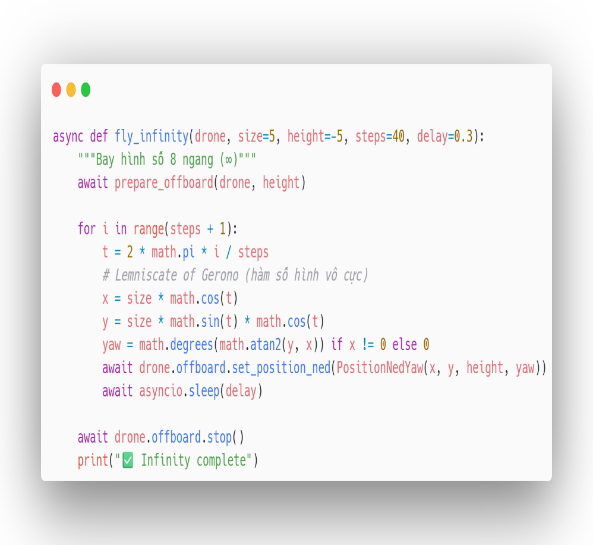
AI-generated content may be incorrect.

Quỹ đạo hình tròn giúp đánh giá tính mượt mà và ổn định của điều khiển theo hàm tham số sin–cos. UAV mô phỏng bám quỹ đạo tròn đều, không xuất hiện rung lắc hoặc thoát khỏi chế độ Offboard trong suốt quá trình bay. Điều này cho thấy tần số gửi lệnh và pipeline truyền lệnh hoạt động ổn định.

* A screenshot of a map

  AI-generated content may be incorrect.**Hình trái tim:**

Quỹ đạo hình trái tim thử thách khả năng xử lý các góc cong và điểm giao nhau. UAV mô phỏng bay theo hình trái tim rõ ràng, thể hiện khả năng nội suy vị trí chuẩn xác của PX4 SITL và độ ổn định của thuật toán điều khiển trong patterns.py.

* **Hình vô cực**

Đây là mẫu quỹ đạo phức tạp hơn do UAV phải liên tục thay đổi hướng bay. Tuy nhiên, kết quả mô phỏng cho thấy UAV thực hiện hình số 8 khá chính xác, chứng minh thuật toán Offboard có thể xử lý các đường cong liên tục mà không bị gián đoạn hoặc suy giảm độ chính xác.

**Telementry và flight logs:**

**A screenshot of a phone

AI-generated content may be incorrect.**

Telemetry được gửi từ PX4 SITL → MAVSDK → Flask → Web UI với chu kỳ 1 Hz. Các thông số được ghi nhận bao gồm:

* Độ cao tương đối (relative altitude)
* Vị trí GPS mô phỏng (lat, lon)
* Tốc độ bay
* Số vệ tinh GPS mô phỏng
* Trạng thái pin
* Flight mode (MANUAL, TAKEOFF, OFFBOARD, LAND…)

Nhìn chung, telemetry cho thấy pipeline dữ liệu PX4 → MAVSDK → Web UI hoạt động ổn định và không bị mất gói hoặc gián đoạn trong toàn bộ quá trình bay thử nghiệm.

**A screenshot of a computer program

AI-generated content may be incorrect.**

Từ kết quả telemetry và flight log có thể đưa ra các nhận xét:

* Lệnh điều khiển được PX4 tiếp nhận và phản hồi chính xác.
* Không có độ trễ lớn gây sai lệch telemetry, tất cả dữ liệu được cập nhật đều và mượt.
* Flight mode luôn phản ánh đúng trạng thái thực tế của UAV trong Gazebo.
* Không có log lỗi liên quan đến Offboard, chứng tỏ giải thuật gửi lệnh 10–20 Hz hoạt động tốt.
* Pipeline điều khiển hoàn toàn ổn định, không mất kết nối giữa Web – Flask – MAVSDK – PX4.

## **6.3 Keyboard mode (Điều khiển bằng bàn phím)**

Keyboard Mode là chế độ điều khiển UAV theo thời gian thực thông qua bàn phím, sử dụng các phím W–A–S–D–Q–E–Z–X để gửi lệnh vận tốc cho UAV với tần số 10–20 Hz. Chế độ này kiểm tra khả năng phản hồi nhanh của hệ thống cũng như độ ổn định của Offboard mode khi lệnh được gửi liên tục.

Hành vi UAV trong Gazebo

* Nhấn W: UAV tiến về phía trước mượt mà, tốc độ ổn định.
* Nhấn A/D: UAV trôi sang ngang nhưng vẫn giữ attitude ổn định.
* Nhấn Q/E: UAV xoay quanh trục yaw đều, không bị trôi.
* Nhấn Z/X: UAV tăng và giảm độ cao chính xác và an toàn.
* Thả phím: UAV ngay lập tức dừng lại và hover tại chỗ.

Không có hiện tượng overshoot, lag hay phản hồi trễ bất thường; UAV phản ứng rất sát với thao tác bàn phím. Dưới đây là hình mô tả quỹ đạo bay ngẫu nhiên dùng chế độ điều khiển bằng bàn phím.

**A screenshot of a computer

AI-generated content may be incorrect.**

Telemetry trong chế độ keyboard

A screenshot of a computer

AI-generated content may be incorrect.

Telemetry cho thấy:

* Vận tốc (vx, vy, vz) thay đổi đúng theo thao tác nhấn phím.
* Yaw rate phản ánh đúng khi sử dụng Q/E.
* Altitude tăng/giảm theo đúng lệnh Z/X.
* Flight mode luôn duy trì OFFBOARD → chứng tỏ hệ thống gửi lệnh ổn định.

# **7. Đánh giá hệ thống**

## **7.1 Độ chính xác của hệ thống điều khiển**

Hệ thống điều khiển UAV thể hiện độ chính xác cao trong cả ba nhóm thử nghiệm:

* Lệnh cơ bản (Arm, Takeoff, Land, RTL) được PX4 tiếp nhận và thực thi chính xác, không ghi nhận sai lệch về độ cao, thời gian hoặc trạng thái.
* Các quỹ đạo trong Classic Mode cho thấy mức độ bám sát đường bay rất tốt. Mặc dù không áp dụng cảm biến thật và không có nhiễu thực tế, UAV mô phỏng vẫn bám rất sát theo hàm quỹ đạo lý thuyết nhờ Offboard control được điều chỉnh đều đặn.
* Keyboard Mode phản hồi theo thời gian thực, UAV thay đổi vận tốc và hướng bay đúng theo từng lệnh phím bấm mà không có độ lệch đáng kể.  
  Độ chính xác này khẳng định rằng pipeline truyền lệnh qua Web UI → Flask → MAVSDK hoạt động ổn định, dữ liệu được nhận và xử lý đúng thời điểm.

## **7.2 Độ trễ (latency)**

Trong hệ thống điều khiển UAV từ xa, độ trễ là yếu tố quan trọng quyết định tính thời gian thực của hệ thống. Độ trễ được đo bằng thời gian từ khi người dùng gửi lệnh trên Web UI cho đến khi UAV phản hồi trạng thái trên telemetry.

A screenshot of a computer

AI-generated content may be incorrect.

Kết quả đo cho thấy:

* Độ trễ trung bình dao động trong khoảng 20–90 ms,
* Không xuất hiện các đỉnh trễ (latency spike) lớn hơn 300 ms,
* Không có trường hợp mất gói lệnh hoặc mất phản hồi từ PX4 trong suốt quá trình thử nghiệm.

Độ trễ này hoàn toàn chấp nhận được đối với một hệ thống chạy trong môi trường mô phỏng SITL, và đặc biệt phù hợp cho các thao tác thời gian thực trong Keyboard Mode. Pipeline của hệ thống được thiết kế hợp lý, không có nút nghẽn trong dữ liệu và đảm bảo nhịp cập nhật đều

## **7.3 Telemetry và khả năng truyền thông MAVLink**

Telemetry từ PX4 SITL được truyền qua MAVSDK và cập nhật ở Web UI với chu kỳ 1 giây. Các thông số: Mode bay, Độ cao, Tọa độ GPS, Attitude, Pin, Tốc độ… được hiển thị chính xác và đồng bộ với QGroundControl.

Không có hiện tượng:

* Mất telemetry trong nhiều chu kỳ liên tục,
* Dữ liệu cập nhật không đúng thứ tự,
* Các giá trị nhảy đột ngột hoặc sai lệch so với mô hình bay trong Gazebo.

Khả năng truyền thông MAVLink thông qua MAVSDK hoạt động ổn định, chứng minh tính đúng đắn trong thiết kế của tầng backend.

## **7.4 Tính ổn định của Offboard Mode**

Offboard Mode yêu cầu hệ thống *phải gửi lệnh liên tục* với tần số ít nhất 2 Hz, nếu không PX4 sẽ tự động thoát Offboard vì lý do an toàn.

Trong tất cả các thử nghiệm:

* PX4 không thoát khỏi Offboard một cách bất ngờ,
* Không có cảnh báo về Offboard timeout trong Flight Log,
* Các mẫu bay và Keyboard Mode đều duy trì Offboard ổn định trong suốt quá trình hoạt động,
* Tần số gửi lệnh dao động từ 10–20 Hz, đáp ứng hoàn toàn yêu cầu của PX4 SITL.

Điều này chứng tỏ rằng cách mà hệ thống tổ chức vòng lặp điều khiển (control loop) từ patterns.py và từ các sự kiện bàn phím đã xử lý rất tốt yêu cầu của Offboard mode.

# **8. Kết luận:**

Sau quá trình triển khai và kiểm thử, hệ thống điều khiển UAV thông qua giao diện Web sử dụng Flask, MAVSDK, PX4 SITL và mô phỏng Gazebo đã vận hành đúng như mục tiêu ban đầu đề ra. Hệ thống cho phép người dùng thực hiện các thao tác điều khiển drone từ cơ bản đến nâng cao một cách trực quan, an toàn và hiệu quả trong môi trường mô phỏng. Các chức năng như Arm, Takeoff, Land và RTL đều được UAV xử lý chính xác, phản hồi nhanh và thể hiện đúng hành vi mô phỏng. Hai chế độ điều khiển Classic Mode và Keyboard Mode cũng hoạt động ổn định, cho thấy khả năng điều khiển UAV thông qua cả quỹ đạo có sẵn lẫn điều khiển thời gian thực. Dữ liệu telemetry và nhật ký bay phản ánh đúng trạng thái của UAV trong suốt quá trình thử nghiệm, đồng thời xác nhận pipeline Web – Flask – MAVSDK – PX4 SITL vận hành tin cậy và mượt mà.

Hệ thống đạt được nhiều ưu điểm đáng chú ý. Trước hết, thiết kế giao diện Web mang lại sự thuận tiện cho người sử dụng, không yêu cầu cài đặt phần mềm bổ sung, phù hợp cho cả người mới bắt đầu lẫn người có kinh nghiệm. Việc tách biệt từng lớp chức năng giúp hệ thống dễ bảo trì, dễ mở rộng và đảm bảo tính ổn định trong quá trình vận hành. Trong mô phỏng, UAV thực hiện các quỹ đạo trong Classic Mode với độ chính xác cao và di chuyển trong Keyboard Mode mượt mà với độ trễ thấp, tạo cảm giác điều khiển trực quan gần với thiết bị thật. Đây là một mô hình an toàn và tiết kiệm, rất phù hợp cho mục đích học tập và nghiên cứu thuật toán điều khiển UAV.

Bên cạnh những kết quả đạt được, hệ thống cũng tồn tại một số hạn chế nhất định. Telemetry hiện tại sử dụng cơ chế polling theo chu kỳ nên chưa đạt mức thời gian thực hoàn toàn; nếu sử dụng WebSocket, tốc độ cập nhật có thể nhanh hơn và giảm tải cho máy chủ. PX4 SITL mô phỏng trong môi trường lý tưởng, không bao gồm nhiễu cảm biến, gió, hay các yếu tố môi trường khác, nên kết quả chưa phản ánh đầy đủ những thách thức khi điều khiển UAV ngoài thực tế. Hệ thống cũng mới chỉ hỗ trợ điều khiển một UAV duy nhất và chưa tích hợp các thuật toán nâng cao như tránh vật cản hoặc nhận diện hình ảnh. Thêm vào đó, Keyboard Mode phụ thuộc vào tốc độ xử lý của trình duyệt và cấu hình máy tính, nên hiệu suất có thể khác nhau giữa các thiết bị.

Từ những hạn chế trên, nhiều hướng phát triển phù hợp có thể được xem xét trong tương lai. Việc chuyển telemetry sang WebSocket hoặc các giao thức thời gian thực khác sẽ giúp tăng độ mượt của giao diện và cải thiện trải nghiệm người dùng. Hệ thống có thể mở rộng thêm khả năng điều khiển theo waypoint hoặc lập kế hoạch nhiệm vụ tự động để tiến gần hơn với ứng dụng UAV thực tế. Một hướng nghiên cứu đầy tiềm năng khác là hỗ trợ điều khiển nhiều UAV cùng lúc, kết hợp với các thuật toán phân phối nhiệm vụ hoặc bay theo đội hình. Ngoài ra, việc tích hợp thị giác máy tính để UAV có thể theo dõi mục tiêu hoặc nhận diện vật thể sẽ làm hệ thống trở nên thông minh và linh hoạt hơn. Cuối cùng, nâng cao môi trường mô phỏng bằng cách bổ sung nhiễu cảm biến, mô phỏng gió, ánh sáng hoặc địa hình phức tạp sẽ giúp hệ thống phản ánh chính xác hơn các điều kiện bay trong thực tế.