TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

NGHIÊN CỨU TRẠM MẶT ĐẤT ỨNG DỤNG GIAO TIẾP VÀ QUẢN LÝ THIẾT BỊ BAY

Sinh viên thực hiện: TRẦN MINH HIẾU

Lớp ĐT01 – K60

Giảng viên hướng dẫn: PGS. TS. NGUYỄN HOÀNG HẢI

TS. NGUYỄN ANH QUANG

Hà Nội, 6-2020

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ - VIỄN THÔNG

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

NGHIÊN CỨU TRẠM MẶT ĐẤT ỨNG DỤNG GIAO TIẾP VÀ QUẢN LÝ THIẾT BỊ BAY

Sinh viên thực hiện: TRẦN MINH HIẾU

Lớp ĐT01 – K60

Giảng viên hướng dẫn: PGS. TS. NGUYỄN HOÀNG HẢI

TS. NGUYỄN ANH QUANG

Cán bộ phản biện:

Hà Nội, 6-2020

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho giảng viên hướng dẫn)

Tên giảng viên đánh giá: .....

Họ và tên sinh viên: .....MSSV:.....

Tên đồ án: .....

.....

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:
Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) *(chọn 1 trong 3 trường hợp)*						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế	5				
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest)	2				
10c	Không có thành tích về nghiên cứu khoa học	0				
Điểm tổng		**/50**				
Điểm tổng quy đổi về thang 10						

Nhận xét khác *(về thái độ và tinh thần làm việc của sinh viên)*

.....

.....

.....

.....

.....

.....

Ngày: … / … / 20…

Người nhận xét

(Ký và ghi rõ họ tên)

ĐÁNH GIÁ QUYỂN ĐỒ ÁN TỐT NGHIỆP

(Dùng cho cán bộ phản biện)

Giảng viên đánh giá:.....

Họ và tên sinh viên:.....MSSV:.....

Tên đồ án: .....

.....

Chọn các mức điểm phù hợp cho sinh viên trình bày theo các tiêu chí dưới đây:
Rất kém (1); Kém (2); Đạt (3); Giỏi (4); Xuất sắc (5)

Có sự kết hợp giữa lý thuyết và thực hành (20)						
1	Nêu rõ tính cấp thiết và quan trọng của đề tài, các vấn đề và các giả thuyết (bao gồm mục đích và tính phù hợp) cũng như phạm vi ứng dụng của đồ án	1	2	3	4	5
2	Cập nhật kết quả nghiên cứu gần đây nhất (trong nước/quốc tế)	1	2	3	4	5
3	Nêu rõ và chi tiết phương pháp nghiên cứu/giải quyết vấn đề	1	2	3	4	5
4	Có kết quả mô phỏng/thực nghiệm và trình bày rõ ràng kết quả đạt được	1	2	3	4	5
Có khả năng phân tích và đánh giá kết quả (15)						
5	Kế hoạch làm việc rõ ràng bao gồm mục tiêu và phương pháp thực hiện dựa trên kết quả nghiên cứu lý thuyết một cách có hệ thống	1	2	3	4	5
6	Kết quả được trình bày một cách logic và dễ hiểu, tất cả kết quả đều được phân tích và đánh giá thỏa đáng	1	2	3	4	5
7	Trong phần kết luận, tác giả chỉ rõ sự khác biệt (nếu có) giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai	1	2	3	4	5
Kỹ năng viết quyển đồ án (10)						
8	Đồ án trình bày đúng mẫu quy định với cấu trúc các chương logic và đẹp mắt (bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến; căn lề thống nhất, có dấu cách sau dấu chấm, dấu phảy v.v.), có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn đúng quy định	1	2	3	4	5
9	Kỹ năng viết xuất sắc (cấu trúc câu chuẩn, văn phong khoa học, lập luận logic và có cơ sở, từ vựng sử dụng phù hợp v.v.)	1	2	3	4	5
Thành tựu nghiên cứu khoa học (5) *(chọn 1 trong 3 trường hợp)*						
10a	Có bài báo khoa học được đăng hoặc chấp nhận đăng/Đạt giải SVNCKH giải 3 cấp Viện trở lên/Có giải thưởng khoa học (quốc tế hoặc trong nước) từ giải 3 trở lên/Có đăng ký bằng phát minh, sáng chế	5				
10b	Được báo cáo tại hội đồng cấp Viện trong hội nghị SVNCKH nhưng không đạt giải từ giải 3 trở lên/Đạt giải khuyến khích trong các kỳ thi quốc gia và quốc tế khác về chuyên ngành (VD: TI contest)	2				
10c	Không có thành tích về nghiên cứu khoa học	0				
Điểm tổng		**/50**				
Điểm tổng quy đổi về thang 10						

Nhận xét khác của cán bộ phản biện

.....

.....

.....

.....

.....

.....

Ngày: … / … / 20…

Người nhận xét

(Ký và ghi rõ họ tên)

LỜI NÓI ĐẦU

Ngành hàng không hiện nay có sự phát triển nhanh chóng về số lượng, kiểu loại, tính chất hoạt động bay. Nhiều loại hình hoạt động bay mới xuất hiện đã góp phần làm cho lĩnh vực hàng không ngày càng phong phú, đáp ứng đầy đủ các yêu cầu của đời sống, xã hội, phát triển kinh tế, văn hóa của nhân loại và dần trở thành lĩnh vực giao thông đóng vai trò then chốt. Loại hình hoạt động bay không người lái tuy mới phát triển mạnh mẽ trong khoảng vài năm trở lại đây nhưng đã đặt ra nhiều thách thức trong việc bảo đảm an toàn bay chung do các quy định hiện hành đối với loại hình hoạt động bay này còn chưa được hoàn thiện.

Drone hay còn gọi là unmanned aerial vehicle (UAV) là những thiết bị bay không người lái có thể điều khiển từ xa hoặc lập trình trước. Drone có cấu tạo gồm các phần chính: bộ động cơ, vi xử lý trung tâm, cánh quạt, giá đỡ, nguồn. Drone có thể di chuyển trên cao và bay xa tới hàng chục km.

Ban đầu, chúng được dùng chủ yếu cho mục đích quân sự, như bay trinh sát với khả năng chụp không ảnh, truyền hình ảnh về căn cứ chỉ huy, hay tìm diệt những mục tiêu khó tiếp cận bằng vũ khí mang theo. Dần dần UAV ngày càng được sử dụng sâu rộng cho mục đích dân sự, từ việc đáp ứng thú chơi tiêu khiển của người dùng, cho đến giao hàng hóa, rải phân, tưới cây chăm sóc mùa màng, theo dõi đàn gia súc, giám sát rừng, vườn thú hoang dã, quay phim, chụp ảnh từ trên cao, cứu hộ cứu nạn những nơi hiểm trở…

Để thực hiện được nhiều hành động và nhiệm vụ khó khăn như vậy, thì phần mềm điều khiển drone chiếm một vai trò cực kỳ quan trọng. Các nhà phát triển trên toàn thế giới vẫn đang miệt mài nghiên cứu để có thể xây dựng các phần mềm điều khiển ngày càng hoàn thiện hơn. Đó cũng là một chủ đề rất thú vị và đầy thách thức đối với một sinh viên như em. Chính vì vậy, để có thể tìm hiểu cũng như thực hành xây dựng một phần mềm điều khiển drone, em quyết định chọn chủ đề: *Nghiên cứu trạm mặt đất ứng dụng giao tiếp và quản lý thiết bị bay*.

Đặc biệt, em xin gửi lời cảm ơn tới PGS. TS. Nguyễn Hoàng Hải, TS. Nguyễn Anh Quang và các bạn trong Lab C9-417 đã tận tình chỉ bảo, hướng dẫn cũng như tạo điều kiện để em có thể có môi trường làm đồ án tốt nhất trong suốt thời gian vừa qua. Em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Tôi là Trần Minh Hiếu, mã số sinh viên 20151368, sinh viên lớp Điện tử 01, khóa K60. Người hướng dẫn là PGS. TS. Nguyễn Hoàng Hải và TS. Nguyễn Anh Quang. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án *Nghiên cứu trạm mặt đất ứng dụng giao tiếp và quản lý thiết bị bay* là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

Hà Nội, ngày 15 - 06 - 2020

Người cam đoan

TRẦN MINH HIẾU

MỤC LỤC

DANH MỤC KÍ HIỆU VÀ TỪ VIẾT TẮT

- UAV: Unmanned aerial vehicle – thiết bị bay không người lái.

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

TÓM TẮT ĐỒ ÁN

Việc tìm hiểu, nghiên cứu và xây dựng phần mềm giao tiếp và quản lý thiết bị bay là cực kỳ quan trọng, góp phần không nhề nhỏ trong sự phát triển vượt bậc của công nghệ bay không người lái trong thời đại hiện nay. Trên thế giới có rất nhiều các tổ chức, cá nhân nghiên cứu về chủ đề này, dẫn đến một lợi thế cho các nhà phát triển là có rất nhiều các tài liệu để tham khảo. Tuy nhiên, với những người mới bắt đầu tìm hiểu, đó lại là một thách thức lớn khi không biết bắt đầu tìm hiểu từ đâu, và chọn loại tài liệu nào để nghiên cứu. Chính vì vậy, trong phạm vi đồ án *Nghiên cứu trạm mặt đất ứng dụng giao tiếp và quản lý thiết bị bay*, em sẽ trình bày một cách tổng quan nhất về các bước tiếp cận công nghệ bay không người lái, cũng như tìm hiểu và xây dựng phần mềm giao tiếp và quản lý thiết bị bay trên phần mềm mô phỏng.

Phần mềm em mô phỏng trong đồ án này đã thực hiện một số chức năng cơ bản như: lấy được thông tin về tọa độ, độ cao của drone; ra lệnh cho drone thực hiện hành động cất cánh, cài đặt nhiệm vụ bay cho drone bằng cách nhập tọa độ trực tiếp và chọn điểm bay trên bản đồ. Hy vọng đề tài đồ án của em sẽ giúp cho những bạn muốn tìm hiểu và nghiên cứu về công nghệ bay không người lái sẽ có cái nhìn tổng quan hơn, một cách tiếp cận dễ dàng hơn, và sẽ là đòn bẩy để các bạn có thể thực hiện được cách công cuộc nghiên cứu sau này.

ABSTRACT

The research and development of communication software and flight management software is extremely important, contributing significantly to the rapid development of unmanned aerial technology in the present era. In the world, there are many organizations and individuals researching on this topic, leading to an advantage for developers that there is a lot of reference material. For beginners, however, it is a big challenge not knowing where to start learning, and what kind of material to research. Therefore, in the scope of the research project of ground station application of communication and management of aircraft, I will present an overview of the steps to approach unmanned technology, as well as learn about and building communication software and flight management software on the simulator.

The software I simulated in this project has performed some basic functions such as obtaining information about the coordinates, the altitude of the drone; order drones to take off, set flight missions for drones by entering coordinates directly and selecting a flight point on the map. Hopefully, your project will help but you want to learn and research on drone technology will have an overview, an easier approach, and will be leverage for you to the research can be done later.

TỔNG QUAN

Chủ đề xây dựng phần mềm giao tiếp và quản lý thiết bị bay tuy không quá mới mẻ nhưng để tiếp cận với nó có khá nhiều vấn đề. Đầu tiên ta phải tìm hiểu tổng quan về phân loại và cấu trúc của thiết bị bay. Bên cạnh đó, hiện nay có rất nhiều phần mềm giao tiếp và quản lý thiết bị bay có sẵn trên thị trường, chúng ta có thể tham khảo về tính năng, cách thức hoạt động của chúng. Tiếp theo chúng ta cần tìm hiểu về giao thức giao tiếp của thiết bị bay và trạm mặt đất, trong đồ án này em sử dụng giao thức The Micro Air Vehicle Link (MAVLink), cụ thể là sử dụng thư viện MAVSDK được phát triển từ MAVLink với API cho C++. Cuối cùng cũng là phần quan trọng nhất, đó là áp dụng các kiến thức kể trên để mô phỏng một trạm mặt đất giao tiếp và quản lý thiết bị bay. Tuy nhiên do thời gian có hạn nên phần mềm mô phỏng sẽ tập trung vào một số chức năng chính như: cập nhật thông tin về tọa độ, vị trí của thiết bị bay; điều khiển thiết bị bay thực hiện một số hành động như cất cánh và hạ cánh, cài đặt nhiệm vụ cho thiết bị bay bằng cách nhập tọa độ trực tiếp và chọn điểm bay trên bản đồ. Nội dung nghiên cứu cụ thể sẽ được em trình bày tại 3 chương dưới đây:

Chương 1: Tổng quan về drone và trạm mặt đất.

Chương 2: Giao thức MAVLink và thư viện MAVSDK.

Chương 3: Xây dựng trạm mặt đất giao tiếp và quản lý thiết bị bay.

CHƯƠNG 1. TỔNG QUAN VỀ DRONE VÀ TRẠM MẶT ĐẤT

Các thiết bị bay không người lái rất đa dạng, từ những loại tên lửa hành trình đến những chiếc máy bay phục vụ cho nhiệm vụ trắc địa bản đồ hay gần gũi nhất là những chiếc flycam để quay phim chụp ảnh. Bên cạnh đó cũng có rất nhiều phần mềm điều khiển (trạm mặt đất) để có thể giao tiếp và quản lý thiết bị bay. Để có thể hiểu rõ về thiết bị và trạm mặt đất, dưới đây em xin trình bày định nghĩa, phân loại các thiết bị UAV và tìm hiểu nguyên lý hoạt động cơ bản cũng như cấu tạo của một loại UAV phổ biến nhất hiện nay đó chính là loại sử dụng bốn động cơ, với phần mềm điều khiển là QGroundControl.

1.1 Tổng quan về drone

1.1.1 UAV là gì?

UAV là viết tắt của "Unmanned Aerial Vehicle" ("Phương tiện hàng không không người lái", hay thường gọi là "Máy bay không người lái"). UAV đôi khi cũng được dùng để chỉ các hệ thống UAVS (Unmanned Aerial Vehicle System - Hệ thống phương tiện bay không người lái) hoặc UAS (Unmanned Aerial System - Hệ thống máy bay không người lái). Ủy ban Quản lý Hàng không Liên bang Hoa Kỳ (FAA) đang sử dụng cụm từ UAS để nhấn mạnh rằng các hệ thống này không chỉ bao gồm máy bay mà còn bao gồm cả trạm kiểm soát trên mặt đất và một số thiết bị/yếu tố khác.

Đúng với tên gọi của mình, trên thiết bị UAV hoàn toàn không có phi công lái. UAV được điều khiển từ xa từ một trạm điều khiển trên mặt đất có thể tự bay theo lịch trình được lặp trình sẵn của các hệ thống máy tính phức tạp. Máy bay không người lái có những hình dạng và kích cỡ khác nhau nên mỗi loại sẽ phù hợp với từng nhu cầu sử dụng khác nhau. UAV có thể được thu hồi tái sử dụng hoặc không, có thể mang theo tải trọng hoặc không tùy vào từng mục đích cụ thể.

1.1.2 Ứng dụng của UAV

Ban đầu được biết đến với mục đích sử dụng quân sự, tuy nhiên với sự phát triển vượt bậc của công nghệ bay không người lái, máy bay không người lái hiện đang được sử dụng trong rất nhiều lĩnh vực khác nhau bởi những lợi ích và ưu điểm tuyệt vời của nó. Dưới đây là một số ứng dụng chính của drone:

- **Tìm kiếm và cứu hộ**: drone rất hữu ích trong việc tìm kiếm và giải cứu. Ví dụ, chúng được sử dụng trong chữa cháy để xác định số lượng khí cụ thể trong khí (CO, CO2, và tương tự) sử dụng thiết bị đo đặc biệt.

- **An ninh**: nhiều cơ quan sử dụng máy bay không người lái để bảo vệ con người trong các trường hợp khẩn cấp khác nhau. Ví dụ, họ có thể giúp phối hợp một loạt các hoạt động bảo mật và có thể giữ nguyên bằng chứng.
- **Kiểm tra**: nhiều hệ thống như đường dây điện, tua-bin gió và đường ống có thể được kiểm tra bằng máy bay không người lái.
- **Giám sát**: một drone cho phép ghi lại và theo dõi từ bầu trời, và do đó, chúng phù hợp để giám sát các sự kiện công cộng, các cuộc biểu tình hoặc bất kỳ sự kiện đáng ngờ nào mà khó có thể kiểm soát trên diện rộng. Một công cụ tuyệt vời cho cảnh sát!
- **Khoa học và nghiên cứu**: họ giúp các nhà khoa học rất nhiều trong các công trình nghiên cứu để quan sát sự xuất hiện khác nhau trong tự nhiên hoặc một môi trường đặc biệt từ bầu trời. Ví dụ, máy bay không người lái được sử dụng để ghi lại các cuộc khai quật khảo cổ, trong các tai nạn hạt nhân (đo ô nhiễm), trong giám sát sông băng, quan sát một vụ phun trào núi lửa, …
- **Chụp ảnh & quay video trên không**: với máy bay không người lái được trang bị camera HD, chúng ta có thể chụp những bức ảnh hấp dẫn và quay cảnh có chất lượng tuyệt vời từ bầu trời.
- **Khảo sát & gis (ánh xạ)**: sử dụng máy ảnh đa phổ và máy quét laze, máy bay không người lái có thể tạo bản đồ 3D chất lượng cao. Do đó, họ đã tìm thấy các ứng dụng trong các lĩnh vực khác nhau, bao gồm viễn thám, khảo sát và lập bản đồ, …
- **Hệ thống chở hàng không người lái**: drone cũng phục vụ trong việc vận chuyển các gói hàng nhẹ. Bằng cách này, chúng ta có thể vận chuyển hàng hóa an toàn, thân thiện với môi trường và nhanh chóng bằng đường hàng không.
- **Dự báo thời tiết**: máy bay không người lái đang được phát triển để theo dõi thời tiết nguy hiểm và không thể đoán trước. Vì chúng rẻ và không người lái, máy bay không người lái có thể được gửi vào bão và lốc xoáy , để các nhà khoa học và dự báo thời tiết có được những hiểu biết mới về hành vi và quỹ đạo của chúng. Các cảm biến chuyên dụng của nó có thể được sử dụng để chi tiết các thông số thời tiết, thu thập dữ liệu và ngăn ngừa rủi ro.
- **Giám sát động vật hoang dã**: máy bay không người lái đã phục vụ như là một răn đe đối với những kẻ săn trộm. Chúng cung cấp sự bảo vệ chưa từng có cho động vật, như voi, tê giác và mèo lớn, một mục tiêu ưa thích của những kẻ săn trộm. Với máy ảnh nhiệt và cảm biến, máy bay không người lái có khả năng hoạt động vào ban đêm. Điều này cho phép họ theo dõi và nghiên cứu về động vật hoang dã mà không gây ra bất kỳ sự xáo trộn nào và cung cấp cái nhìn sâu sắc về mô hình, hành vi và môi trường sống của chúng.

Ngoài ra còn rất nhiều các ứng dụng khác của drone đã và đang được áp dụng trong cuộc sống thực tiễn. Chính vì lợi ích to lớn mà nó đem lại nên nghiên cứu và phát triển drone là một chủ đề được rất nhiều các cá nhân, tổ chức quan tâm và đầu tư phát triển.

1.1.3 Phân loại drone

Hiện nay có rất nhiều loại UAV, có thể phân loại chúng dựa theo một số lượng lớn các đặc tính hiệu suất. Các khía cạnh như là trọng lượng, độ bền, phạm vi, tốc độ và tải trọng cánh là các thông số kỹ thuật quan trọng giúp phân biệt các loại UAV. Dưới đây em xin trình bày một số cách phân loại các thiết bị UAV.

1.1.3.1 Phân loại theo trọng lượng

Các thiết bị UAV có dải trọng lượng rộng, có thể chỉ nặng vài tram gram đến vài chục tấn.

- Đầu tiên là các UAV "siêu nặng", là những chiếc có trọng lượng cất cánh trên 2 tấn. Loại này sẽ bao gồm X - 45, Darkstar, Predator B và Global Hawk.
- Tiếp theo là các loại UAV nặng từ 200 đến 2000 kg. Loại này sẽ bao gồm các thiết bị từ Outrider đến Fire Scout
- Phân loại thứ 3 sẽ là các UAV có trọng lượng trung bình từ 50 đến 200 kg. Loại này bao gồm các thiết bị từ Raven đến Phoenix
- Loại thứ 4 sẽ là phân khúc các UAV có trọng lượng nhẹ. Từ 5 đến 50 kg.
- Cuối cùng là các micro UAV. Loại này sẽ có trọng lượng dưới 5 kg. Loại này chúng ta được thấy nhiều hơn trong dân sự.

1.1.3.2 Phân loại theo độ bền và phạm vi

Một biện pháp phân loại hữu ích khác cho UAV là phân loại theo độ bền và phạm vi. Hai thông số này quan hệ mật thiết với nhau, tỉ lệ thuận với nhau.

- Bay cao, là tất cả các UAV có thể bay trên độ cao 10000m. Các loại này đa phần là các loại UAV hạng nặng như: X - 45, Darkstar, Predator B và Global Hawk.
- Độ cao bay trung bình, là các loại UAV có độ cao bay từ 1000 đến 10000 m.
- Độ cao bay thấp, là các loại UAV bất kì có độ cao bay nhỏ hơn 1000 m.

1.1.3.3 Phân loại theo độ cao

Theo độ cao, các loại UAV có thể phân thành 3 mức như sau:

- Bay cao, là tất cả các UAV có thể bay trên độ cao 10000m. Các loại này đa phần là các loại UAV hạng nặng như: X - 45, Darkstar, Predator B và Global Hawk.
- Độ cao bay trung bình, là các loại UAV có độ cao bay từ 1000 đến 10000 m.
- Độ cao bay thấp, là các loại UAV bất kì có độ cao bay nhỏ hơn 1000 m.

1.1.3.4 Phân loại theo tải trọng cánh

Các loại UAV cũng có thể được phân loại theo tải trọng cánh. Tải trọng cánh được tính bằng cách lấy tổng khối lượng UAV chia cho diện tích cánh quạt.

- Đối với UAV cỡ nhỏ, tải trọng cánh nằm trong khoảng dưới 50 kg/m^2.
- Các loại UAV tầm trung có tải trọng cánh từ 50 đến 100 kg/m^2.
- Các loại UAV cỡ lớn có tải trọng cánh lớn hơn 100 kg/m^2.

1.1.3.5 Phân loại theo động cơ

Mỗi một loại UAV thực hiện mỗi nhiệm vụ khác nhau. Vì vậy cần những động cơ khác nhau để có thể thực hiện nhiệm vụ này.

- Động cơ điện: được sử dụng ở những chiếc UAV cỡ vừa và nhỏ phục vụ mục đích nghiên cứu.
- Động cơ sử dụng piston chủ yếu được sử dụng ở những chiếc UAV hạng nặng phục vụ nhiều trong quân sự.

1.1.3.6 Phân loại theo kích thước

Theo kích thước, UAV được chia thành những loại sau:

- UAV rất nhỏ: Lớp UAV rất nhỏ áp dụng cho các UAV có kích thước từ kích thước của một con côn trùng lớn đến dài 30-50 cm. Các UAV giống như côn trùng, có cánh hoặc cánh xoay, là một thiết kế vi mô phổ biến. Chúng có kích thước cực kỳ nhỏ, trọng lượng rất nhẹ và có thể được sử dụng cho gián điệp và chiến tranh sinh học. Những chiếc lớn hơn sử dụng cấu hình máy bay thông thường. Sự lựa chọn giữa cánh vỗ hoặc cánh quay là một vấn đề về khả năng cơ động mong muốn. Thiết kế dựa trên cánh vỗ cho phép đậu và hạ cánh trên các bề mặt nhỏ. Ví dụ về các UAV rất nhỏ là Muỗi IAI Malat của Israel (với sải cánh dài 35 cm và độ bền 40 phút), Aurora của Hoa Kỳ (với sải cánh 60 cm và dài 33 cm), CyberQuad của Úc Mini (với 42x42 cm vuông) và model mới nhất của họ, CyberQuad Maxi.
- UAV nhỏ: Lớp UAV nhỏ (đôi khi còn được gọi là UAV mini) áp dụng cho các UAV có ít nhất một chiều lớn hơn 50 cm và không lớn hơn 2 mét.
- UAV trung bình: Lớp UAV trung bình áp dụng cho các UAV quá nặng để một người mang theo nhưng vẫn nhỏ hơn một chiếc máy bay hạng nhẹ. Chúng thường có sải cánh khoảng 5-10 m và có thể mang trọng tải từ 100 đến 200 kg. Ví dụ về các UAV cánh cố định trung bình là Hunter Israel-Mỹ và Watchkeeper Anh. Có những thương hiệu khác được sử dụng trong quá khứ, như Boeing Eagle Eye của Mỹ, RQ-2 Pioneer, hệ thống BAE Skyeye R4E và RQ-5A Hunter. Hunter có sải cánh dài 10,2 m và dài 6,9 m. Nó nặng khoảng 885 kg khi cất cánh. RS-20 của American Aerospace là một ví dụ khác về UAV chéo, bao gồm các thông số kỹ

thuật của UAV cỡ nhỏ và trung bình. Nhiều UAV trung bình khác có thể được tìm thấy trong bài tập đọc. Ngoài ra còn có một số UAV cỡ trung bình quay.

- UAV lớn: Lớp UAV lớn được sử dụng chủ yếu cho các hoạt động chiến đấu của quân đội. Ví dụ về các UAV lớn này là General Atomics Predator A và B của Hoa Kỳ và Northrop Grumman Global Hawk của Hoa Kỳ.

1.1.4 Cấu tạo của thiết bị bay loại sử dụng 4 động cơ

Như đã tìm hiểu ở phần trên thì chúng ta có thể biết được hiện tại UAV có rất nhiều chủng loại khác nhau. Mỗi loại thiết bị UAV lại có cấu tạo và nguyên lý hoạt động khác nhau. Để nghiên cứu hết được chúng trong điều kiện hiện nay gần như là không thể. Vì vậy, trong phạm vi đồ án này em sẽ tìm hiểu và nghiên cứu loại UAV cỡ nhỏ sử dụng 4 động cơ cánh quạt phản lực hay còn gọi là drone.

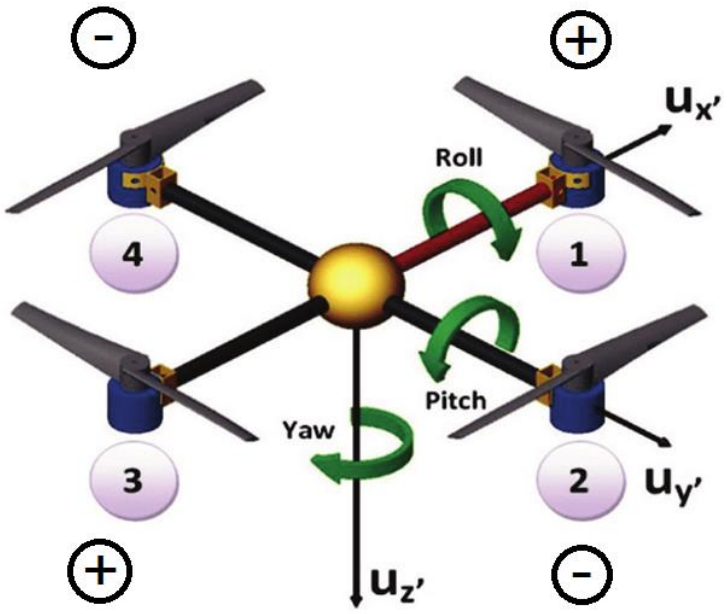
Hình 1.1 Drone phổ biến hiện nay – Flycam

Về cơ bản, drone gồm các thành phần chính: vi mạch tích hợp bộ xử lý, động cơ, nguồn cấp năng lượng (pin), 4 cánh quạt. Điều khiển bay bằng bộ điều khiển từ xa hoặc/và lập trình sẵn theo lộ trình, tọa độ dựa trên GPS. Nhiều drone hiện nay đã được tích hợp GPS nên luôn định vị được đang bay ở đâu.

Bộ điều khiển drone thường sử dụng sóng radio tần số 2,4GHz, hình thức không khác mấy so với những bộ điều khiển từ xa của máy bay mô hình truyền thống, gồm hai nút bấm và ăng ten có thể gấp gọn. Một số bộ điều khiển có sự kết hợp cả tín hiệu 2,4GHz và Wi-Fi, trông giống tay cầm điều khiển máy chơi game hoặc chúng có thể dựa trên ứng dụng điều khiển chạy trên smartphone hay máy tính bảng.

1.1.5 Nguyên lý hoạt động của drone bốn động cơ

Muốn làm cho một chiếc Drone 4 rotors có thể bay được thì chúng ta cần phải hiểu được nguyên lý hoạt động của nó. Sau đây chúng ta sẽ đi tìm hiểu nguyên lý hoạt động của một quadcopter – một loại máy bay sử dụng 4 cánh quạt phản lực.



Hình 1.2 Hướng quay cánh quạt của Drone

Hiện nay, các công nghệ về máy bay không người lái nói chung hay loại drone sử dụng quadcopters nói riêng đều rất hiện đại nhưng tất cả các loại quadcopters đều sử dụng một nguyên lý cũ đó là sử dụng lực đẩy của các dòng khí để nâng máy bay lên cũng như để chuyển hướng bay. Nguyên lý hoạt động chính của mô hình này hoạt động dựa trên sự chuyển động của các dòng khí do cánh máy bay tạo ra di chuyển xuống dưới làm vật bay lên trên và sự điều chỉnh vận tốc từng động cơ sẽ làm thay đổi hướng bay của Quadcopter.

Để cho một quadcopter bay lên không trung, một lực phải được tạo ra, bằng hoặc lớn hơn lực hấp dẫn. Bây giờ, quadcopters sử dụng thiết kế động cơ và hướng cánh quạt cho động cơ đẩy kiểm soát cơ bản lực hấp dẫn đối với quadcopter. Việc quay các cánh quạt quadcopter đẩy không khí xuống. Tất cả các lực tác dụng theo cặp (Định luật thứ ba của Newton), có nghĩa là đối với mọi lực tác động có đều có phản lực bằng (về độ lớn) và ngược lại (theo hướng). Do đó, khi roto tác dụng một lực với không khi theo hướng cùng chiều với lực hút trái đất thì không khí đẩy ngược roto với một lực bằng lực roto tác động nhưng ngược hướng. Khi lực này lớn hơn lực hút trái đất thì Drone bắt đầu có thể rời khỏi mắt đất. Với nguyên lý này một chiếc Drone đã có thể thực hiện được 3 nhiệm vụ: lên thẳng, lơ lửng và đáp xuống mặt đất.

Xét hệ tọa độ Body (một hệ tọa độ có tâm trùng với tâm của drone, trục x trùng với hướng bay của drone, trục y thuộc mặt phẳng bay của drone vuông góc với trục x và hướng bên phải trục x. Trục z vuông góc với mặt phẳng bay của drone và hướng xuống dưới).

Trước khi tìm hiểu chiều quay của quadcopter chúng ta sẽ xem xét các thuật ngữ sau:

- Yaw: Đây là xoay hoặc xoay đầu của quadcopter hoặc sang phải hoặc trái quanh trục z. Đó là chuyển động cơ bản để quay quadcopter.
- Pitch: Đây là chuyển động của quadcopter về phía trước và phía sau, chuyển động xoay quanh trục y.
- Roll: Là chuyển động bay nghiêng về một bên, chuyển động này xoay quanh trục x dùng để điều hướng drone sang phải hoặc trái.

Xét tại thời điểm Drone lơ lửng (thời điểm có tổng động lượng góc bằng 0 và các mô men xoắn tại mỗi đông cơ phải có độ lớn bằng nhau) gán giá trị cho các động cơ 1, 2, 3, 4 các động lượng góc lần lượt là: +4, -4, +4, -4. Ta sẽ xem xét các trạng thái chuyển đông của drone dưới đây.

1.1.5.1 Chuyển động Yaw

Cặp cánh quạt 1, 3 quay thuận chiều kim đồng hồ, trong khi đó cặp cánh 2, 4 lại quay ngược chiều kim đồng hồ nhằm cân bằng động lượng góc cho cả quadcopter. Cả 4 cánh phải sinh ra một moment xoắn bằng nhau khi quadcopter cất cánh và hạ cánh hay khi ở trạng thái lơ lửng. Để có được chuyển động yaw ta cần thay đổi vận tốc góc của 2 cặp cánh 1, 3 và 2, 4 nhưng vẫn phải đảm bảo 2 động cơ trực đối có mô men xoắn bằng nhau (để tránh quadcopter rơi vào trạng thái pitch hay roll). Cụ thể như sau:

Khi thay đổi động lượng góc của động cơ 1 và 3 còn +3, lúc đó tổng động lượng góc của quadcopter sẽ là -2. Quadcopter sẽ có xu hướng quay theo chiều kim đồng hồ nhằm mục đích tạo ra động lượng góc +2 để cân bằng với động lượng góc -2 mà động cơ gây ra. Ứng dụng điều này ta được quadcopter ở trạng thái yaw (xoay bên phải).

Tương tự, khi thay đổi động lượng góc của động cơ 2 và 4 còn -3, lúc đó tổng động lượng góc của quadcopter sẽ là +2. Quadcopter sẽ có xu hướng quay ngược chiều kim đồng hồ nhằm mục đích tạo ra động lượng góc -2 để cân bằng với động lượng góc +2 mà động cơ gây ra. Ứng dụng điều này ta được quadcopter ở trạng thái yaw (xoay bên trái).

1.1.5.2 Chuyển động pitch

Theo giả thiết ban đầu quadcopter đang ở trạng thái lơ lửng. Để quadcopter có được chuyển động pitch ta cần thay đổi mô men xoắn của các cặp động cơ 1, 3. Khi mô men xoắn của động cơ 1 nhỏ hơn mô men xoắn của động cơ 3, đồng thời giữ nguyên mô men xoắn của cặp động cơ 2, 4. Khi đó quadcopter sẽ nghiêng về phía trước kèm theo chuyển động tiến về phía trước. Nếu sự chênh lệch giữa mô men xoắn của cặp động cơ 1, 3 đủ lớn, quadcopter sẽ chuyển động nhào lộn. Tương tự, khi mô men xoắn của động cơ 1 lớn hơn động cơ 3. Khi đó quadcopter sẽ nghiêng về phía sau kèm theo chuyển động lùi. Sự nhào lộn xảy ra khi độ chênh lệch mô men đủ lớn.

1.1.5.3 Chuyển động roll

Theo giả thiết ban đầu quadcopter đang ở trạng thái lơ lửng. Để quadcopter có được chuyển động roll ta cần thay đổi mô men xoắn của các cặp động cơ 2, 4. Khi mô men xoắn của động cơ 2 nhỏ hơn mô men xoắn của động cơ 4, đồng thời giữ nguyên mô men xoắn của cặp động cơ 1, 3. Khi đó quadcopter sẽ nghiêng về phía bên phải kèm theo chuyển động rẽ phải. Nếu sự chênh lệch giữa mô men xoắn của cặp động cơ 2, 4 đủ lớn, quadcopter sẽ chuyển động nhào lộn. Tương tự, khi mô men xoắn của động cơ 2 lớn hơn động cơ 3. Khi đó quadcopter sẽ nghiêng về phía trái kèm theo chuyển động rẽ trái. Sự nhào lộn xảy ra khi độ chênh lệch mô men đủ lớn.

1.2 Tổng quan về trạm mặt đất

Về cơ bản, thiết bị bay không người lái được điều khiển từ xa bằng cách sử dụng một thiết bị điều khiển vô tuyến (trạm điều khiển mặt đất) hoặc bay tự động bằng cách sử dụng máy lái tự động liền thân máy. Máy lái tự động có thể đi theo con đường bay được lập trình sẵn thông qua xử lý dữ liệu về vị trí và định vị từ hệ thống định vị vệ tinh toàn cầu (GNSS) và thiết bị thu và đo quán tính toàn cầu (IMU). Trạm điều khiển mặt đất nhận dữ liệu về chuyến bay như vị trí thiết bị bay, độ cao, tốc độ và trạng thái pin thông qua kết nối vô tuyến từ xa. Người vận hành thiết bị giám sát dữ liệu chuyến bay trong quá trình thiết bị đang bay và có thể có can thiệp thủ công bất cứ lúc nào.

Hiện nay, có rất nhiều trạm mặt đất được thiết kế dưới dạng phần mềm, chúng ta có thể tải và cài đặt chúng trên máy tính, điện thoại và sử dụng một cách dễ dàng. Hầu hết các phần mềm đều có các chức năng cơ bản như: nhận dữ liệu về tọa độ, độ cao, thông số của drone; có thể ra lệnh cho drone thực hiện hành động cất cánh và hạ cánh, hay thiết lập các nhiệm vụ cho drone. Tuy nhiên có một số phần mềm được thiết kế giúp chúng ta can thiệp sâu hơn vào hệ thống như thiết lập các thông số của drone, điều chỉnh phương thức giao tiếp với drone. Ở mục này em sẽ giới thiệu về QGroundControl - một phần mềm điều khiển mã nguồn mở với rất nhiều tính năng để chúng ta có thể tham khảo.

1.2.1 Yêu cầu thiết kế của trạm mặt đất

Trạm mặt đất chứa giao diện người dùng để trực quan hóa một số thông số điều khiển và dữ liệu từ xa từ máy bay không người lái. Trạm mặt đất vẫn được liên kết trực tiếp với mô-đun từ xa để liên lạc với drone. Trạm mặt đất trình bày dữ liệu telemetry đến từ máy bay không người lái như vị trí GPS, độ cao, vận tốc dọc, vận tốc ngang, chế độ máy bay, v.v. trong một giao diện duy nhất, xác định trạng thái hiện tại của máy bay không người lái. Trạm mặt đất cũng chứa một số cài đặt để sửa đổi hiệu chuẩn cảm biến drone từ IMU, hiệu chuẩn từ kế, hiệu chỉnh radio, chế độ máy bay, v.v.

1.2.1.1 Yêu cầu chức năng cho trạm mặt đất

Thiết kế của trạm mặt đất cho UAV nên giải quyết một số yêu cầu chức năng cơ bản như được liệt kê dưới đây:

Điều khiển máy bay: chức năng này đề cập đến khả năng kiểm soát và bay UAV hiệu quả trong nhiệm vụ của mình. Nó ngụ ý sự sẵn có của các liên kết dữ liệu phù hợp để liên lạc với UAV với khả năng bản đồ hóa vị trí và lệnh của nó.

Kiểm soát tải trọng: các UAV mang nhiều loại cảm biến cần được vận hành từ mặt đất. Điều này sẽ yêu cầu các điều khiển khác nhau dành riêng cho tải trọng được mang, liên kết dữ liệu phù hợp để thu thập dữ liệu, thiết bị hiển thị và phương tiện lưu trữ để sử dụng thông tin thu được.

Lập kế hoạch nhiệm vụ: trạm mặt đất sẽ hỗ trợ bộ điều khiển UAV lập kế hoạch cho nhiệm vụ có tính đến các khả năng và giới hạn của các hệ thống UAV. Hàm này cung cấp cho bộ điều khiển các đầu vào dựa trên kiến thức cần thiết đến một hồ sơ nhiệm vụ tối ưu mà không gây nguy hiểm cho sự an toàn của UAV.

Phân tích dữ liệu và truyền tin: nói chung, thông tin hình ảnh thu được từ payload cần xử lý, chẳng hạn như khả năng hiển thị, ngụy trang, v.v. Ngoài ra, thông tin cụ thể, chẳng hạn như vị trí và kích thước của các mục tiêu quan tâm, cần được tính toán từ các hình ảnh. Các khả năng này được cung cấp bởi một hệ thống con xử lý hình ảnh (có khả năng giao tiếp cần thiết) để phân tích và trích xuất thông tin và phổ biến dữ liệu cho người dùng cuối cùng.

Chẩn đoán hệ thống/máy bay: theo quan điểm của các chức năng được cung cấp bởi hệ thống UAV, cả trạm mặt đất và UAV đều có xu hướng là các hệ thống phức tạp đòi hỏi cơ sở thử nghiệm tự động để bảo trì và triển khai hiệu quả.

Phân tích sau chuyến bay: khả năng lưu trữ dữ liệu chuyến bay và dữ liệu tải trọng và phân tích tương tự sau chuyến bay là một yêu cầu thiết yếu của UAV và trạm mặt đất.

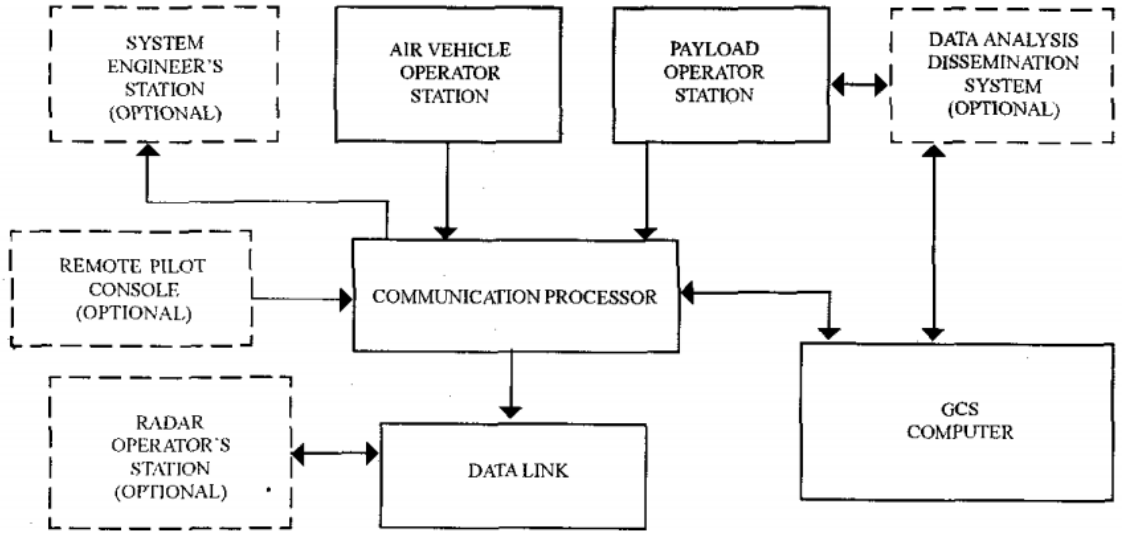
1.2.1.2 Yêu cầu hoạt động của trạm mặt đất

Trạm mặt đất được yêu cầu phải đáp ứng các yêu cầu hoạt động sau đây để tối đa hóa tiện ích của chúng:

Khả năng hoạt động chung: kiến trúc trạm mặt đất sẽ cho phép vận hành hệ thống UAV ở chế độ tích hợp với các nền tảng khác, chẳng hạn như máy bay có người lái, vệ tinh, v.v.

Khả năng tương tác: khi có nhiều hơn một loại UAV đang hoạt động, khả năng sử dụng cùng một trạm mặt đất (thông qua cấu hình lại phần mềm) với các loại UAV khác nhau là rất cần thiết, để đạt được lợi thế về chi phí hệ thống và đào tạo người vận hành.

Kiến trúc mở: hệ thống trạm mặt đất sẽ kết hợp các hệ thống con có các mô-đun và giao diện tiêu chuẩn công nghiệp. Cách tiếp cận này có thể xử lý các vấn đề, chẳng hạn như lỗi thời công nghệ và cho phép tăng cường các hệ thống con riêng lẻ mà không ảnh hưởng đến các phần khác của trạm mặt đất.



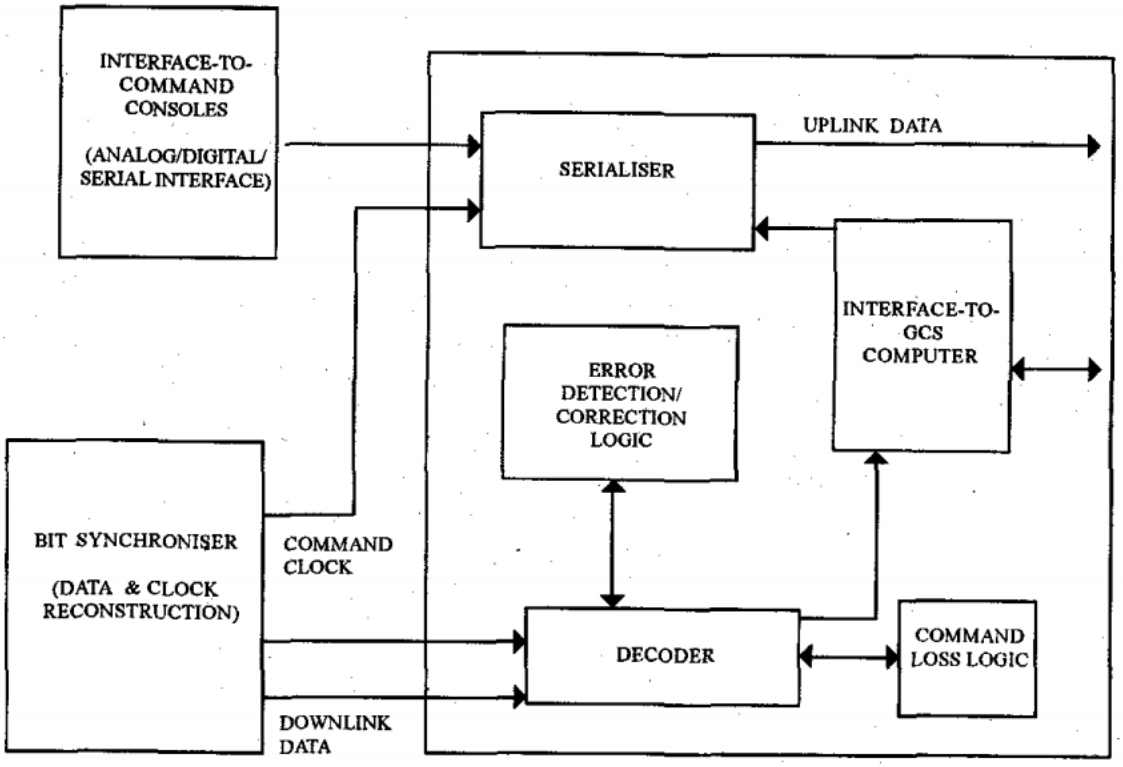
Hình 1.3 Sơ đồ khối của trạm mặt đất

1.2.1.3 Quy trình giao tiếp

Luồng dữ liệu giữa trạm mặt đất và UAV được quản lý bởi một bộ các hệ thống xử lý giao tiếp chuyên dụng ở hai đầu. Các chức năng chính của bộ xử lý thông tin là:

- Mã hóa các lệnh theo tỷ lệ lấy mẫu được chỉ định.
- Giải mã lỗi dữ liệu đường xuống.
- Khả năng xử lý lỗi một cách rõ ràng.
- Xác thực dữ liệu trước khi nó được gửi để xử lý thêm.

Bộ xử lý giao tiếp kết hợp hai đơn vị chức năng là bộ mã hóa và bộ giải mã. Hệ thống con bộ mã hóa có phần cứng thu thập dữ liệu và bộ nối tiếp kênh. Hệ thống con bộ giải mã có bộ đồng bộ hóa bit để trích xuất đồng bộ dữ liệu, hệ thống giải mã để giải mã dữ liệu riêng lẻ và giao diện phù hợp với bộ xử lý UAV/trạm mặt đất để trao đổi thông tin. Hệ thống giải mã có logic cần thiết để xác định các điều kiện mất liên kết. Hệ thống xử lý dựa trên bộ xử lý cung cấp khả năng xử lý tin nhắn/tệp cần thiết để tải lên các kế hoạch nhiệm vụ lên UAV. Sơ đồ khối của bộ xử lý thông tin trạm mặt đất được hiển thị trong Hình 1.4



Hình 1.4 Sơ đồ khối của bộ xử lý thông tin trạm mặt đất

1.2.2 Tổng quan về QGroundControl [1]

QGroundControl cung cấp đầy đủ các chức năng điều khiển bay và thiết lập hệ thống cho các phương tiện chạy bằng PX4 hoặc ArduPilot. Nó cung cấp việc sử dụng dễ dàng và đơn giản cho người mới bắt đầu, trong khi vẫn cung cấp hỗ trợ tính năng cao cấp cho người dùng có kinh nghiệm.

Các tính năng chính của QGroundControl:

- Thiết lập/cấu hình đầy đủ của phương tiện chạy bằng ArduPilot và PX4 Pro.
- Hỗ trợ chuyến bay cho các phương tiện chạy PX4 và ArduPilot (hoặc bất kỳ chế độ lái tự động nào khác giao tiếp bằng giao thức MAVLink).
- Lên kế hoạch nhiệm vụ cho chuyến bay tự trị.
- Hiển thị bản đồ chuyến bay cho thấy vị trí xe, đường bay, điểm tham chiếu.
- Truyền phát video.
- Hỗ trợ quản lý nhiều phương tiện.
- QGC chạy trên các nền tảng Windows, OS X, Linux, iOS và Android.

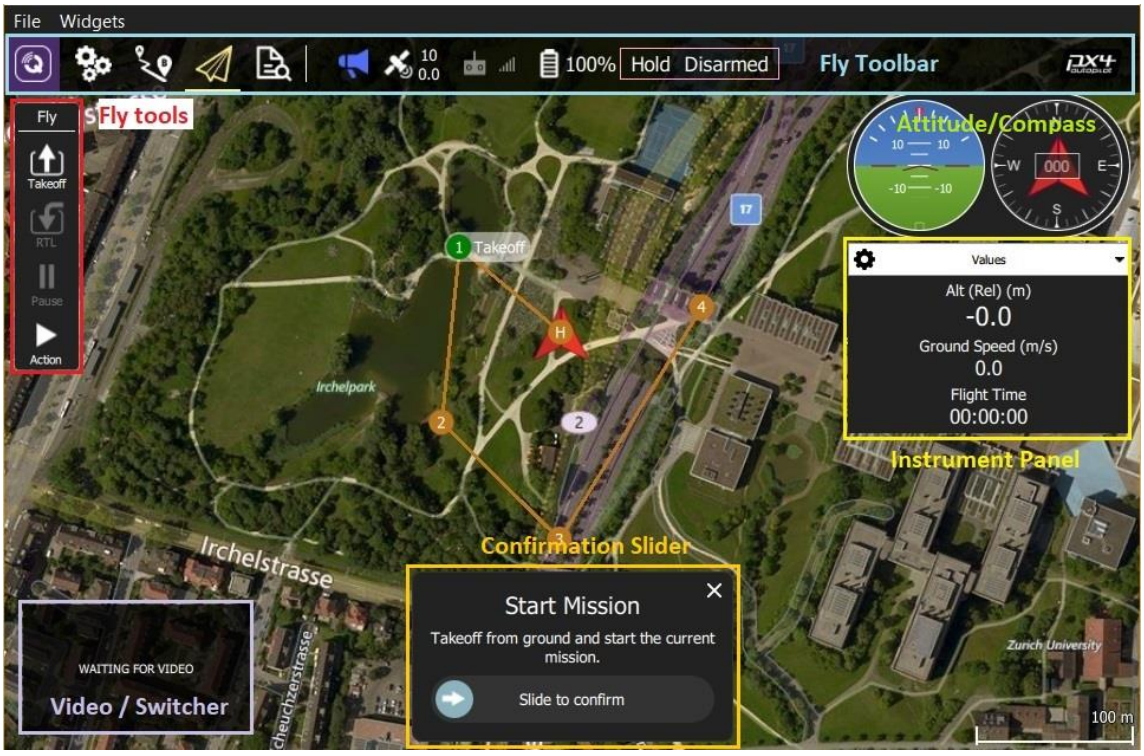
1.2.2.1 Chế độ bay

Fly View được sử dụng để chỉ huy và giám sát phương tiện khi bay.

Chúng ta có thể sử dụng nó để:

- Chạy một danh sách kiểm tra tự động trước chuyến bay.
- Điều khiển nhiệm vụ: bắt đầu, tiếp tục, tạm dừng và tiếp tục .

- Hướng dẫn các phương tiện để sẵn sàng/hủy sẵn sàng/dừng khẩn cấp, cất cánh/hạ cánh, thay đổi độ cao, đi đến hoặc quay quanh một địa điểm cụ thể, và trở lại/RTL .
- Chuyển đổi giữa chế độ xem bản đồ và chế độ xem video (nếu có).
- Hiển thị video, nhiệm vụ, telemetry và các thông tin khác cho phương tiện hiện tại và cũng chuyển đổi giữa các phương tiện được kết nối.



Hình 1.5 Giao diện của QGroundControl

Ảnh chụp màn hình ở trên cho thấy các phần chính của chế độ bay:

- **Map**: hiển thị vị trí của tất cả các phương tiện được kết nối và nhiệm vụ cho phương tiện hiện tại. Chúng ta có thể kéo bản đồ để di chuyển xung quanh (bản đồ sẽ tự động căn giữa lại sau một khoảng thời gian nhất định). Sau khi bay, chúng ta có thể nhấp vào bản đồ để đặt đi tới hoặc xoay vòng tại vị trí.
- **Fly Toolbar**: thông tin trạng thái chính cho các cảm biến (GPS, pin, điều khiển RC) và trạng thái xe (Flight mode, Armed/Disarmed status). Chọn các chỉ số cảm biến để xem chi tiết hơn. Bấm *Fly mode* (ví dụ: "Hold") để chọn chế độ mới. Không phải mọi chế độ có thể có sẵn. Nhấn Armed/Disarmed để chuyển trạng thái sẵn sàng. Nếu bay ta có thể nhấn vào đây để dừng khẩn cấp.
- **Fly tools**: ta có thể sử dụng những công cụ này để:
  - Chuyển đổi giữa cất cánh/hạ cánh.
  - Tạm dừng/khởi động lại hoạt động hiện tại (ví dụ: hạ cánh hoặc nhiệm vụ).
  - Trở lại an toàn (còn được gọi là RTL hoặc Return).

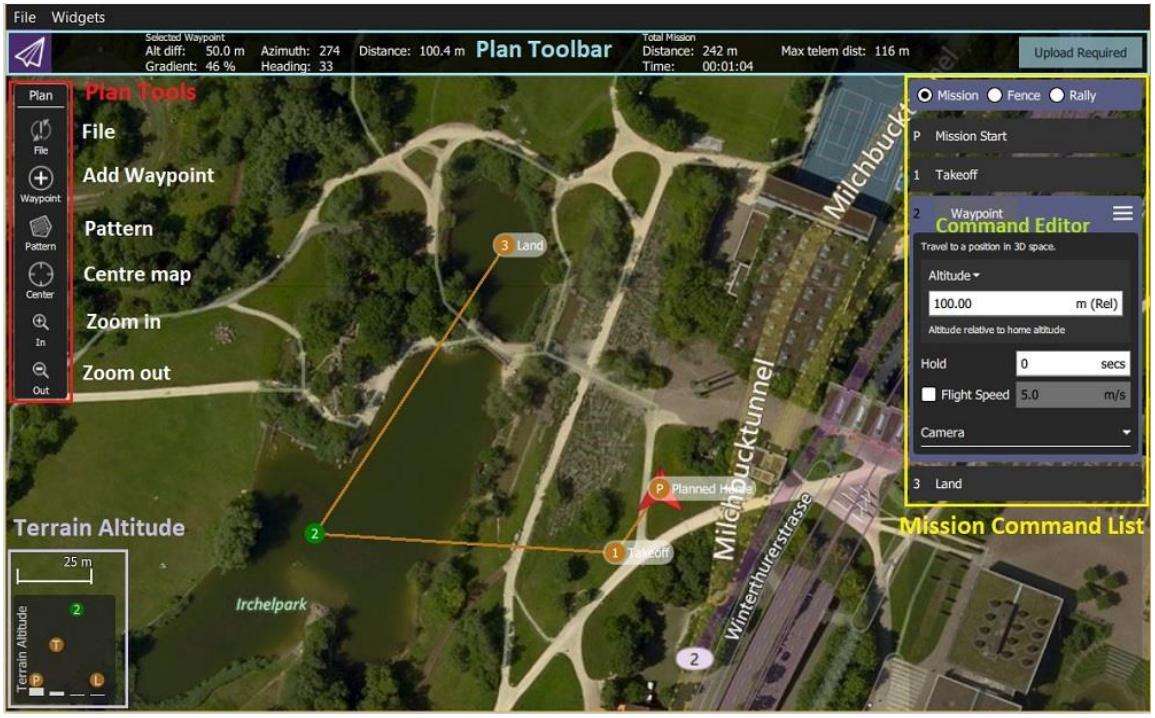
- Nút *Action* đưa ra các tùy chọn thích hợp khác cho tình trạng hiện thời (Confirmation Slider). Các hành động bao gồm thay đổi độ cao hoặc tiếp tục một nhiệm vụ.
- Bật danh sách kiểm tra preflight (tùy chọn công cụ bị tắt theo mặc định).
- **Instrument Panel**: một tiện ích hiển thị thông tin về phương tiện bao gồm: telemetry, camera, video, trạng thái hệ thống và độ rung.
- **Video/Switcher** : chuyển đổi giữa video hoặc bản đồ trong một cửa sổ. Nhấn vào đây để chuyển đổi giữa video và bản đồ. QGroundControl hỗ trợ truyền phát video RTP và RTSP qua kết nối UDP phương tiện. Nó cũng hỗ trợ kết nối trực tiếp hỗ trợ thiết bị UVC. Một Telemetry Overlay được tự động tạo ra như một file phụ đề
- **Confirmation Slider**: thanh trượt để xác nhận các hành động được yêu cầu. Trượt để bắt đầu hoạt động. Nhấn X để hủy.

Có một số phần khác không được hiển thị theo mặc định/chỉ được hiển thị trong một số điều kiện nhất định. Ví dụ: bộ chọn nhiều phương tiện chỉ được hiển thị nếu có nhiều phương tiện và nút công cụ danh sách kiểm tra đèn chiếu sáng chỉ được hiển thị nếu bật cài đặt thích hợp.

1.2.2.2 Chế độ nhiệm vụ

Fly mode được sử dụng để lập kế hoạch cho các nhiệm vụ tự trị cho phương tiện và tải chúng lên phương tiện. Sau khi nhiệm vụ được lên kế hoạch và gửi đến phương tiện, ta chuyển sang *Fly mode* để thực hiện nhiệm vụ.

Nó cũng được sử dụng để định cấu hình GeoFence và Rally Points nếu những phần mềm này được phần mềm hỗ trợ.



Hình 1.6 Chế độ nhiệm vụ

Ảnh chụp màn hình ở trên cho thấy một kế hoạch nhiệm vụ đơn giản bắt đầu bằng một lần cất cánh ở vị trí Planned Home (H), bay qua ba điểm, và sau đó hạ cánh trên điểm cuối cùng (ví dụ: điểm 3).

Các thành phần chính của giao diện là:

- **Map**: hiển thị các chỉ số được đánh số cho nhiệm vụ hiện tại, bao gồm cả Planned. Nhấp vào các chỉ số để chọn chúng (để chỉnh sửa) hoặc kéo chúng xung quanh để định vị lại chúng.
- **Planned Toolbar**: thông tin trạng thái cho điểm tham chiếu hiện được chọn liên quan đến điểm tham chiếu trước đó, cũng như thống kê cho toàn bộ nhiệm vụ (ví dụ: khoảng cách ngang và thời gian cho nhiệm vụ).
  - *Max telem dist* là khoảng cách giữa Planned Home và điểm tham quan xa nhất.
  - Khi được kết nối với một phương tiện, nó cũng hiển thị nút *Upload*, có thể được sử dụng để tải kế hoạch lên phương tiện.
- **Plan Tools**: được sử dụng để tạo và quản lý các nhiệm vụ.
- **Mission Command List/Overlay**: hiển thị danh sách các mục nhiệm vụ hiện tại (chọn các mục để chỉnh sửa).
- **Terrain Altitude Overlay**: hiển thị độ cao tương đối của mỗi lệnh nhiệm vụ.

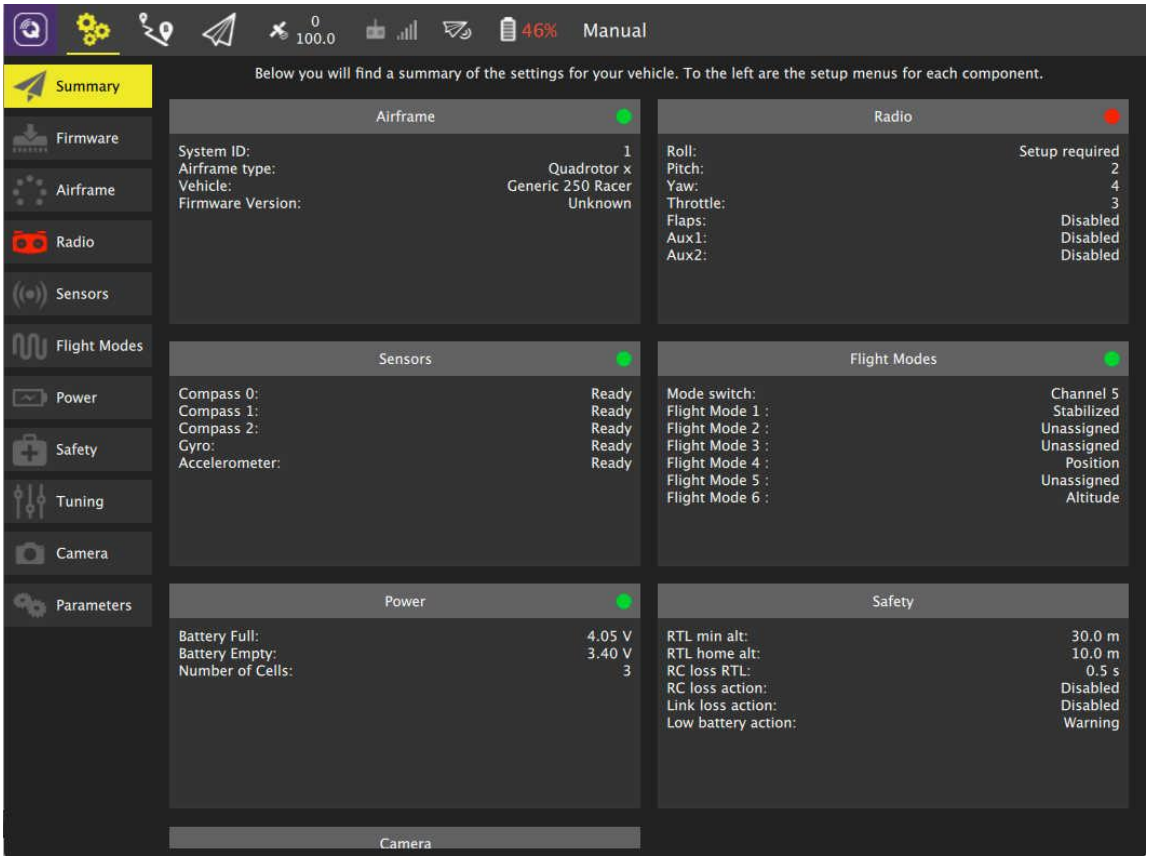
Nó cho ta thấy thông tin liên quan đến điểm tham chiếu hiện tại cũng như số liệu thống kê cho toàn bộ nhiệm vụ.

Ở cấp độ rất cao, các bước để tạo một nhiệm vụ là:

- Thay đổi để xem kế hoạch.
- Thêm điểm tham chiếu hoặc lệnh vào nhiệm vụ và chỉnh sửa khi cần.
- Tải nhiệm vụ lên phương tiện.
- Thay đổi để Fly View và bay nhiệm vụ.

1.2.2.3 Thiết lập hệ thống

Chế độ thiết lập được sử dụng để cấu hình một phương tiện mới trước chuyến bay đầu tiên và/hoặc điều chỉnh một phương tiện được cấu hình.



Hình 1.7 Setup View

Ở bên trái màn hình là tập hợp các tùy chọn thiết lập có sẵn. Nút thiết lập được đánh dấu bằng biểu tượng màu đỏ nếu vẫn còn các cài đặt cần điều chỉnh/chỉ định. Chúng ta không nên bay nếu bất kỳ mục nào trong đây là màu đỏ. Trong ảnh trên, thiết lập Radio chưa hoàn tất.

Danh sách các tùy chọn thiết lập quan trọng cho phương tiện:

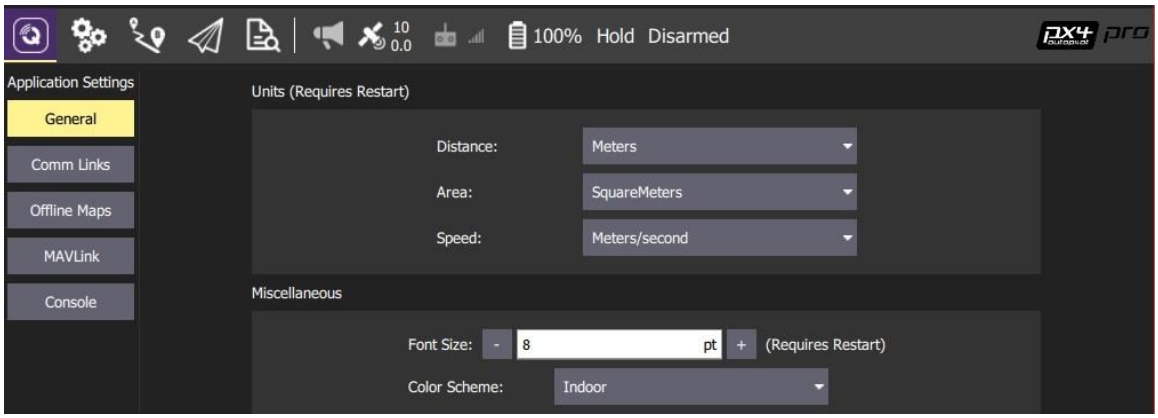
- **Firmware**: flash firmware mới vào phương tiện.
- **Airframe**: chỉ định loại khung máy bay cho phương tiện.
- **Radio**: hiệu chỉnh máy phát điều khiển vô tuyến.
- **Sensors**: hiệu chỉnh các cảm biến trên phương tiện.
- **Flight Modes**: được sử dụng để gán các chế độ máy bay cho các công tắc Máy phát RC.

- **Power**: cài đặt pin và các tùy chọn nguồn bổ sung như hiệu chuẩn ESC.
- **Motors**: thử nghiệm động cơ và thiết lập.
- **Safety**: chỉ định cài đặt cho các tùy chọn liên quan đến an toàn, chẳng hạn như *Return to Home* hoặc *Failsafes*.
- **Tuning**: điều chỉnh thông số bay của phương tiện.
- **Camera**: cấu hình cài đặt cho máy ảnh và gimbal.
- **Parameters**: cho phép sửa đổi tất cả các thông số liên quan đến chiếc phương tiện.

1.2.2.4 Thiết lập ứng dụng

Setup View được sử dụng để cấu hình các thiết lập cho ứng dụng QGroundControl (chứ không phải là một phương tiện cụ thể). Ta không cần phải có một phương tiện được kết nối để thay đổi các giá trị này.

Chúng ta có thể chuyển đổi giữa các tùy chọn cài đặt khác nhau bằng cách nhấp vào các nút ở thanh bên trái.



Hình 1.8 Giao diện thiết lập ứng dụng

Các tùy chọn cài đặt:

- **General**: cài đặt cấu hình ứng dụng chính. Chúng được sử dụng để chỉ định: đơn vị hiển thị, thiết bị tự động kết nối, hiển thị và lưu trữ video, RTK GPS, v.v.
- **Comm Links**: cho phép ta tự tạo các liên kết giao tiếp và kết nối với chúng. Hãy nhớ rằng thông thường điều này là không cần thiết vì QGroundControl sẽ tự động kết nối với các thiết bị phổ biến nhất.
- **Offline Maps**: cho phép lưu trữ bản đồ để sử dụng trong khi không có kết nối Internet.
- **MAVLink**: cài đặt liên quan đến kết nối MAVLink với phương tiện.
- **Console**: được sử dụng để chụp nhật ký ứng dụng để giúp chẩn đoán các sự cố ứng dụng.

Như vậy, ở chương đầu tiên, em đã trình bày được định nghĩa về UAV, cách phân loại UAV và những loại UAV có trên thế giới hiện nay. Em cũng trình bày về

nguyên lý hoạt động của thiết bị không người lái loại sử dụng 4 động cơ và các trạng thái chuyển động của nó. Bên cạnh đó, em đã đưa ra cái nhìn tổng quan về trạm mặt đất và tìm hiểu về QGroundControl – một phần mềm điều khiển mã nguồn mở đang được sử dụng rất phổ biến trên thế giới. Ở chương tiếp theo, em sẽ trình bày về giao thức để liên lạc giữa UAV và trạm mặt đất, đó là giao thức MAVLink và thư viện MAVSDK.

CHƯƠNG 2. GIAO THỨC MAVLINK VÀ THƯ VIỆN MAVSDK

MAVLink là một giao thức liên lạc cho các hệ thống không người lái (ví dụ: máy bay không người lái, robot). Nó chỉ định một tập hợp toàn diện các thông điệp được trao đổi giữa các hệ thống không người lái và trạm mặt đất. Giao thức này được sử dụng trong các hệ thống lái tự động, chủ yếu là ArduPilot và PX4, và cung cấp các tính năng mạnh mẽ không chỉ để giám sát và kiểm soát các nhiệm vụ của hệ thống không người lái, mà còn để tích hợp vào Internet. MAVSDK là thư viện MAVLink với các API cho C ++, iOS, Python và Android. MAVSDK là cách tốt nhất để tích hợp với một hệ thống bay qua MAVLink. Ở trong chương này, em sẽ trình bày một cách tổng quan về giao thức MAVLink và thư viện MAVSDK.

2.1 GIAO THỨC MAVLINK [2]

2.1.1 Tổng quan về MAVLink

MAVLink là một giao thức tuần tự hóa tin nhắn nhẹ được thiết lập tốt dành riêng cho các hệ thống xe không người lái, bao gồm cả máy bay không người lái. Lorenz Meier đã phát hành MAVLink vào năm 2009 theo giấy phép LGPL. MAVLink được thiết kế như một thư viện Marshaling, có nghĩa là nó tuần tự hóa các thông điệp về trạng thái của hệ thống và các lệnh mà nó phải thực hiện thành một định dạng nhị phân cụ thể (tức là một luồng byte), không phụ thuộc vào nền tảng. Bản chất tuần tự hóa nhị phân của giao thức MAVLink làm cho nó nhẹ hơn vì nó có chi phí tối thiểu so với các kỹ thuật tuần tự hóa khác (ví dụ: XML hoặc JSON). Giao tiếp sử dụng MAVLink là hai chiều giữa trạm mặt đất và hệ thống không người lái. Ngoài ra, với tính năng tuần tự hóa nhị phân của MAVLink, các tin nhắn của nó thường có kích thước nhỏ và có thể được truyền qua các phương tiện không dây khác nhau, bao gồm WiFi hoặc thậm chí các thiết bị telemetry nối tiếp với tốc độ dữ liệu thấp. Nó cũng đảm bảo độ tin cậy và tính toàn vẹn của thông điệp bằng cách xác minh tổng kiểm tra kép, trong tiêu đề gói của nó. Tất cả các tính năng này làm cho giao thức MAVLink trở nên phổ biến nhất trong số các đồng nghiệp của nó để liên lạc giữa các hệ thống không người lái và GCS.

Giao thức MAVLink xác định cơ chế về cấu trúc của các thông báo và cách tuần tự hóa chúng ở lớp ứng dụng. Các thông điệp này sau đó được chuyển tiếp đến các lớp thấp hơn (tức là lớp vận chuyển, lớp vật lý) để truyền đến mạng. Ưu điểm của giao thức MAVLink là nó hỗ trợ các loại lớp và phương tiện vận chuyển khác nhau nhờ cấu trúc nhẹ. Nó có thể được truyền qua WiFi, Ethernet (tức là, mạng TCP/IP) hoặc các kênh băng thông thấp từ xa nối tiếp hoạt động ở tần số dưới GHz, cụ thể là 433 MHz, 868 MHz hoặc 915 MHz. Tần số subGHz cho phép đạt được phạm vi liên lạc lớn để điều khiển hệ thống không người lái từ xa. Tốc độ dữ liệu tối đa có thể đạt tới 250 kbps và phạm vi tối đa thường được dự kiến là 500 m, nhưng phụ thuộc nhiều vào môi trường

và mức độ thiết lập nhiễu và ăng ten. Bảng 2.1 trình bày các tính năng của một số thiết bị telemetry thường được sử dụng.

Cách thay thế thứ hai là sử dụng giao diện mạng (network interface), thường là WiFi hoặc Ethernet và truyền phát các tin nhắn MAVLink qua Mạng IP. Trong trường hợp này, hệ thống lái tự động chạy giao thức MAVLink thường hỗ trợ cả kết nối UDP và TCP ở tầng vận chuyển giữa trạm mặt đất và máy bay không người lái, tùy thuộc vào mức độ tin cậy mà ứng dụng yêu cầu. Tất nhiên, người ta thường biết rằng UDP là một giao thức datagram không yêu cầu kết nối giữa máy khách và máy chủ và nó không có cơ chế để đảm bảo rằng các tin nhắn được gửi một cách đáng tin cậy, nhưng cung cấp một sự thay thế trọng lượng nhanh hơn cho thời gian thực và chịu lỗ dòng tin nhắn. Mặt khác, TCP là một giao thức kết nối đáng tin cậy cung cấp độ tin cậy chuyển tốt hơn nhờ cơ chế xác nhận của nó nhưng có thể bị tắc nghẽn và quản lý nặng kết nối. Việc lựa chọn giao thức vận chuyển được để lại cho người dùng tùy thuộc vào yêu cầu anh ta cần cho việc trao đổi tin nhắn giữa hệ thống không người lái và trạm mặt đất.

2.1.2 Các phiên bản tin nhắn và cấu trúc

Hệ thống không người lái liên lạc với trạm mặt đất thông qua việc trao đổi các tin nhắn MAVLink, là các tin nhắn được nối tiếp nhị phân. Tuần tự hóa nhị phân có nghĩa là nội dung của tin nhắn được chuyển thành một chuỗi byte được truyền qua mạng. Người nhận thông điệp nối tiếp thực hiện quá trình giải mã theo hướng ngược lại để khôi phục lại thông điệp ban đầu được gửi. Thuộc tính này của tuần tự hóa nhị phân có một lợi ích đáng kể là giảm kích thước của thông điệp được truyền xuống tối đa so với các loại tuần tự hóa khác, chẳng hạn như XML hoặc thậm chí là JSON có trọng lượng nhẹ hơn. Mỗi tin nhắn MAVLink chứa một tiêu đề được thêm vào tải trọng tin nhắn. Tiêu đề thực hiện thông tin về tin nhắn trong khi tải trọng bao gồm dữ liệu được vận chuyển bởi tin nhắn.

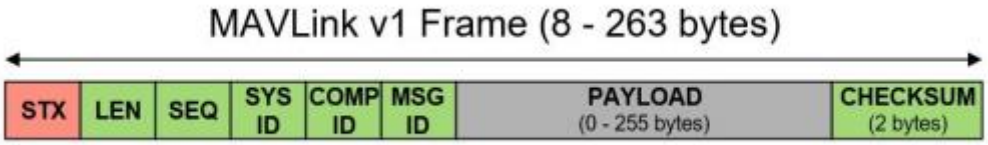
2.1.2.1 MAVLink 1.0 Protocol Header

Trường đầu tiên là STX và dùng để chỉ biểu tượng bắt đầu khung MAVLink. Trong MAVLink 1.0, STX bằng ký hiệu đặc biệt 0XFE. Byte thứ hai (LEN) biểu thị độ dài thông báo theo byte và được mã hóa thành 1 Byte. Byte thứ ba (SEQ) biểu thị số thứ tự của tin nhắn. Nó được mã hóa thành 1 Byte và nhận các giá trị từ 0 đến 255. Khi đạt đến 255, số thứ tự được đặt lại thành 0 và tăng dần trong mỗi thông báo được tạo. Số thứ tự của tin nhắn được kích hoạt để phát hiện mất tin nhắn trong máy thu. SYS byte thứ tư đại diện cho ID hệ thống. Mỗi hệ thống không người lái nên có ID hệ thống, đặc biệt, nếu chúng được quản lý bởi một trạm mặt đất. ID hệ thống 255 thường được phân bổ cho các trạm mặt đất. Một hạn chế của MAVLink 1.0 là nó giới hạn số lượng máy bay không người lái được quản lý bởi một trạm mặt đất là 254 vì ID hệ thống được mã hóa trong 1 Byte. Byte thứ năm là ID thành phần và nó xác định thành phần của hệ thống

đang gửi tin nhắn. Có 27 loại phần cứng (ví dụ như các thành phần) trong MAVLink 1.0. Nếu không có hệ thống con hoặc thành phần, thì nó không được sử dụng. Byte thứ sáu đại diện cho ID tin nhắn (MSGID), dùng để chỉ loại tin nhắn được nhúng trong tải trọng. Ví dụ: ID thông báo bằng 0 đề cập đến một thông báo thuộc loại HEARTBEAT, cho biết hệ thống đang tồn tại và được gửi mỗi giây. Thêm một ví dụ nữa với ID Tin nhắn bằng 33, trong đó đề cập đến một thông báo thực hiện tọa độ GPS của hệ thống không người lái. ID tin nhắn là thông tin cần thiết cho phép phân tích trọng tải và trích xuất thông tin từ nó, dựa trên loại tin nhắn. Mỗi thông báo chứa một số trường được chỉ định và được tuần tự hóa ở định dạng nhị phân theo một thứ tự cụ thể, theo đặc điểm kỹ thuật tiêu chuẩn. Tải trọng được đặt ngay sau ID thông báo và có thể mất tối đa 255 byte. Cuối cùng, hai byte cuối cùng dành cho tổng kiểm tra. CKA và CKB đại diện cho kiểm tra chu kỳ dự phòng (Cyclic Redundancy Check CRC) được tính toán tương ứng với các giá trị hạt giống A và B. CRC đảm bảo rằng thông điệp không bị thay đổi trong quá trình truyền và cả người gửi và người nhận đều có cùng một thông điệp. Nó được tính bằng cách sử dụng hàm băm ITU X.25/SAE AS-4 của các byte của thông báo, ngoại trừ trường STX (hàm băm được áp dụng cho 6 + n + 1 byte và giá trị bổ sung là giá trị nguồn). Giá trị nguồn được thêm vào cuối tin nhắn khi tính toán CRC.

Bảng 2.1 Một so sánh tổng quan giữa các thiết bị telemetry Ardupilot

Telemetry device	Frequency	Range	Voltage	Sensitivity (độ nhạy)	RF transmit power (công suất truyền RF)
Bluetooth	Between 2402 and 2480 MHz, or 2400 and 2483.5 MHz	50m	3.6 to 6V	-80dBm	+4 dBm
SiK Radio v2	900 MHz or 433 MHz	500m	3.3 V	-121dBm	20 dBm
RFD900	900 MHz or 868 MHz	>40 km	3.3 to 5 V	>121 dBm	+30 dBm
Robsence	433 MHz	3-5 km	5 V	-148 dBm	20 dBm



Hình 2.1 Cấu trúc của MAVLink 1.0

Độ dài tin nhắn tối thiểu của MAVLink 1.0 là 8 byte cho các gói xác nhận mà không cần tải trọng. Mặt khác, độ dài tối đa của tin nhắn MAVLink 1.0 là 263 byte cho toàn bộ tải trọng.

2.1.2.2 MAVLink 2.0 Protocol Header

Giao thức MAVLink 2.0 đã được phát hành vào đầu năm 2017 và là phiên bản được đề xuất hiện tại. Nó tương thích ngược với phiên bản MAVLink 1.0 và bao gồm một số cải tiến so với phiên bản MAVLink 1.0.

Tiêu đề MAVLink 2.0 chia sẻ tất cả các trường với tiêu đề MAVLink 1.0 và thêm các trường mới của nó, ngoài việc thay đổi kích thước của một số trường hiện có. Byte đầu tiên là điểm đánh dấu văn bản bắt đầu và giá trị cụ thể của nó là 0xFD cho MAVLink 2.0 (trái ngược với 0xFE cho MAVLink 1.0). Do đó, trình phân tích cú pháp trước tiên phải nhận ra ký tự này trước khi có thể phân tích các trường còn lại của thông báo MAVLink 2.0. Độ dài tải trọng là trường tiếp theo và giống như trong giao thức cũ. MAVLink giới thiệu hai cờ mới trước số thứ tự (SEQ) của tin nhắn. Các cờ đầu tiên là các cờ *không tương thích*, là các cờ ảnh hưởng đến cấu trúc thư. Các cờ cho biết liệu gói có chứa một số tính năng phải được xem xét khi phân tích gói. Ví dụ, cờ *không tương thích* bằng 0x01 có nghĩa là gói được ký và chữ ký được thêm vào cuối gói. Cờ thứ hai là cờ tương thích, không ảnh hưởng đến cấu trúc của tin nhắn. Nó chỉ ra các cờ có thể bị bỏ qua nếu không hiểu và nó không ngăn trình phân tích cú pháp xử lý thông báo ngay cả khi cờ không thể diễn giải được. Ví dụ: điều này có thể đề cập đến các cờ biểu thị mức độ ưu tiên của gói (ví dụ: mức độ ưu tiên cao) vì nó không ảnh hưởng đến cấu trúc gói. Số thứ tự (SEQ), ID hệ thống (SYSID) và ID thành phần (COMPID) giống như trong giao thức MAVLink 1.0. Tuy nhiên, ID tin nhắn (MSGID) được mã hóa thành 24 bit thay vì 8 bit trong phiên bản trước, cho phép số lượng loại tin nhắn trong MAVLink 2.0 cao hơn nhiều, đạt tới 16777215 loại có thể. Không rõ lý do để thiết kế một không gian rộng lớn của các loại thông điệp có thể là gì, vì số lượng khả năng là quá lớn. Trường Payload có thể mất tới 255 byte dữ liệu, tùy thuộc vào loại thông báo. Checksum tương tự như đồng đẳng của nó trong MAVLink 1.0. Cuối cùng, MAVLink 2.0 sử dụng trường Chữ ký tùy chọn gồm 13 byte để đảm bảo rằng liên kết không bị giả mạo. Tính năng này cải thiện đáng kể các khía cạnh bảo mật của MAVLink 1.0 vì nó cho phép xác thực tin nhắn và xác minh rằng nó có nguồn gốc từ một nguồn đáng tin cậy. Chữ ký của tin nhắn được nối thêm nếu các cờ không tương thích được đặt thành 0x01.

13 byte của chữ ký tin nhắn chứa các trường sau:

- **LinkID**: đó là một byte đại diện cho ID của liên kết (kênh) được sử dụng để gửi gói. Một liên kết hoặc một kênh có thể là WiFi hoặc đo từ xa và có thể được kết hợp. Mỗi kênh được sử dụng để gửi dữ liệu nên có LinkID riêng. Nó cung cấp một phương tiện để điều khiển hệ thống không người lái đa kênh bằng MAVLink 2.0.
- **timestamp**: nó được mã hóa với 6 byte theo đơn vị 10 micro giây kể từ ngày 1 tháng 1 năm 2015 GMT. Nó tăng cho mỗi tin nhắn được gửi qua kênh. Nó được áp dụng cho mọi luồng trong đó một luồng được xác định bởi bộ dữ liệu

(SystemID, ElementID, LinkID). timestamp được sử dụng để tránh các cuộc tấn công phát lại.

- **signature**: nó được mã hóa thành 6 byte cho tin nhắn và được tính dựa trên tin nhắn đầy đủ, timestamp và khóa bí mật (secret key). Chữ ký bao gồm 6 byte đầu tiên (48 bites) của hàm băm SHA-256 được áp dụng cho thông báo MAVLink 2.0 (không bao gồm chữ kí, và bao gồm timestamp). Khóa bí mật là khóa đối xứng được chia sẻ gồm 32 byte được lưu trữ ở cả hai đầu, cụ thể là autopilot và trạm mặt đất hoặc API MAVLink.

Hình 2.2 Cấu trúc của MAVLink 2.0

Chữ ký của tin nhắn MAVLink 2.0 có kết quả về cách xử lý tin nhắn MAVLink đến. Nếu tin nhắn được ký, thì nó sẽ bị loại bỏ nếu (i) timestamp của tin nhắn nhận được cũ hơn gói trước nhận được từ cùng một luồng được xác định bởi (SystemID, ElementID, LinkID), chữ ký được tính toán, (ii.) tại chỗ tiếp nhận (at the reception) khác với chữ ký được thêm vào tin nhắn. Điều này có thể suy ra sự thay đổi dữ liệu trong tin nhắn, (iii.) timestamp vượt quá một phút so với timestamp của hệ thống cục bộ. Nếu thông báo không được ký, thì việc chấp nhận/từ chối gói là triển khai cụ thể.

2.1.3 Các loại tin nhắn MAVLink

MAVLink định nghĩa một số loại tin nhắn, được xác định bởi ID tin nhắn của họ. Tin nhắn có ID tin nhắn thấp hơn 255 là phổ biến cho cả MAVLink 1.0 và MAVLink 2.0, những tin nhắn có ID tin nhắn cao hơn 255 là dành riêng cho MAVLink 2.0. Như đã đề cập trong phần trước, ID tin nhắn trong MAVLink 1.0 được mã hóa chỉ trong 8 bit và được mở rộng thành 24 bit trong MAVLink 2.0.

Ta phân loại tin nhắn MAVLink thành hai lớp:

- **Thông báo trạng thái (state messages)**: các tin nhắn này được gửi từ hệ thống không người lái đến trạm mặt đất và chứa thông tin về trạng thái của hệ thống, chẳng hạn như ID, vị trí, vận tốc và độ cao của nó.
- **Thông báo lệnh (command messages)**: chúng được gửi từ trạm mặt đất (hoặc chương trình người dùng) đến hệ thống không người lái để thực hiện một số hành động hoặc nhiệm vụ bằng autopilot. Ví dụ, trạm mặt đất có thể gửi lệnh cho máy bay không người lái cất cánh hoặc hạ cánh hoặc đi đến một điểm dừng hoặc thậm chí là để thực hiện một nhiệm vụ với một số điểm dừng.

Số lượng tin nhắn MAVLink là rất lớn, để có thể mô tả hết được các dạng tin nhắn trong phạm vi bài báo cáo gần như là không thể. Em sẽ trình bày các các thông điệp lệnh và trạng thái có liên quan nhất được sử dụng trong các triển khai chung của autopilots.

2.1.3.1 Thông báo trạng thái

Có một số loại thông điệp trạng thái được xác định trong MAVLink.

HEARTBEAT message: HEARTBEAT là message quan trọng nhất trong MAVLink và cấu trúc của nó được mô tả trong Hình 2.3. Nó cho biết hệ thống phương tiện có mặt và hoạt động. Hệ thống không người lái định kỳ gửi heartbeat message đến trạm mặt đất để thông báo cho GCS rằng nó đang hoạt động. The heartbeat là một thông điệp cần thiết. Ngoài tiêu đề, message payload chứa thông tin cần thiết về hệ thống không người lái. Trường đầu tiên là loại, nó cho biết loại Micro Aerial Vehicle. Theo thông số kỹ thuật mới nhất, có 33 loại được xác định trước được xác định trong MAV_TYPE, bao gồm tứ giác (MAV_TYPE_QUADROTOR = 2), máy bay trực thăng (MAV_TYPE_HELICOPTER = 4), cánh cố định (MAV_TYPE_FIX_ WING = 1), và các loại khác. Trường autopilot cho biết loại lái tự động. Có một số loại được định nghĩa trong cấu trúc liệt kê MAV_AUTOPILOT. Ví dụ: MAV_AUTOPILOT_GENERIC = 0 biểu thị chế độ lái tự động chung, MAV_AUTOPILOT_ARDUPILOTMEGA = 3 biểu thị chế độ lái tự động ArduPilot, MAV_AUTOPILOT_PX4 = 12 cho chế độ tự động PX4.

type	autopilot	base_mode	custom_mode	system_status	mavlink_version
8 bits	8 bits	8 bits	32 bits	8 bits	8 bits

Hình 2.3 MAVLink Heartbeat Message

Trường base_mode cho biết các chế độ hoạt động khác nhau. Hiểu cơ sở_mode rất quan trọng để phân tích chính xác heartbeat message và trích xuất thông tin hữu ích từ nó. Nó được mã hóa trong 8 bit. Có 8 cờ được xác định trước, từ 2^0 đến 2^7.

Dưới đây là tám chế độ khác nhau:

- Flag = 1 được dành riêng để sử dụng trong tương lai.
- Flag = 2 có nghĩa là chế độ kiểm tra được bật. Chế độ này được sử dụng cho các thử nghiệm tạm thời và không được sử dụng cho các chuyến bay thông thường.
- Flag = 4 có nghĩa là chế độ tự trị (AUTO) được bật. Điều này có nghĩa là hệ thống không người lái đang hoạt động tự chủ bằng cách điều hướng đến các điểm mục tiêu được gửi đến từ trạm mặt đất. Trong chế độ AUTO, một nhiệm vụ được tải vào chế độ lái tự động. Một nhiệm vụ bao gồm một tập hợp nhiều điểm tham chiếu mà hệ thống phải điều hướng.

- Flag = 8 có nghĩa là chế độ GUIDED được bật. Trong chế độ GUIDED, một nhiệm vụ bao gồm một điểm duy nhất được gửi đến hệ thống. Hệ thống sau đó điều hướng đến vị trí được chỉ định một cách tự động.
- Flag = 16 có nghĩa là hệ thống ổn định thái độ (định hướng và độ cao) và có thể là vị trí của nó, bằng điều khiển tự động. Điều này đòi hỏi các cảm biến bên ngoài như GPS trong môi trường trong nhà, cảm biến độ cao (barometer, LIDAR) hoặc chụp chuyển động để định vị trong nhà để có thể di chuyển trong trạng thái ổn định. Hệ thống cần các đầu vào điều khiển bên ngoài để làm cho nó di chuyển xung quanh.
- Flag = 32 có nghĩa là phần cứng trong trình giả lập vòng lặp được kích hoạt, tức là, tất cả các động cơ và bộ truyền động của động cơ đều bị chặn trong khi chế độ lái tự động bên trong hoạt động hoàn toàn.
- Flag = 64 có nghĩa là chế độ thủ công được bật, yêu cầu phi công điều khiển thủ công hệ thống bằng cách sử dụng đầu vào điều khiển từ xa. Trong điều khiển thủ công, không có điều khiển tự động được thực hiện bởi chế độ lái tự động.
- Flag = 128 hệ thống ở trạng thái vũ trang (armed state), có nghĩa là động cơ được bật/chạy và có thể bắt đầu bay.

The custom_mode cũng rất cần thiết. Nó chỉ ra các cờ cụ thể tự động được giải thích ngoài chế độ cơ sở. Nó được sử dụng trong phân tích thông điệp nhịp tim để xác định các chế độ bay của hệ thống lái tự động. Có các giá trị được xác định trước cho chế độ tùy chỉnh bao gồm 0 cho chế độ bay thủ công, 4 cho chế độ được hướng dẫn, 10 cho chế độ tự động, 11 cho chế độ RTL, 9 cho chế độ LAND, 2 cho ALT_HOLD và 5 cho LOITER.

Trường system_status đại diện cho một cờ biểu thị trạng thái hệ thống. Có các trạng thái được xác định theo thông số kỹ thuật mới nhất:

- system_status = 0 đề cập đến một hệ thống không phải là hệ thống khởi tạo hoặc trạng thái không xác định.
- system_status = 1 chỉ ra rằng hệ thống đang khởi động.
- system_status = 2 có nghĩa là hệ thống đang thực hiện hiệu chuẩn. Trên thực tế, hiệu chuẩn cảm biến là một giai đoạn rất quan trọng để đảm bảo rằng các cảm biến bay như Đơn vị đo lường quán tính (IMU) và La bàn đều phù hợp và chạy như mong đợi.
- system_status = 3 có nghĩa là hệ thống ở chế độ chờ và có thể khởi động bất cứ lúc nào.
- system_status = 4 chỉ ra rằng các động cơ đã hoạt động và hệ thống đang hoạt động và có thể là trên không.

- system_status = 5 chỉ ra các lỗi tiềm ẩn và hệ thống đang ở trạng thái quan trọng, mặc dù nó vẫn có thể điều hướng. Điều này có thể xảy ra ví dụ trong khi nhiễu tạm thời hoặc mức pin bắt đầu thấp, v.v.
- system_status = 6 điều này có nghĩa là tình huống khẩn cấp trong đó hệ thống không người lái bị mất kiểm soát đối với một số bộ phận và đang trong tình trạng gặp sự cố. Hệ thống có thể đã bị sập.
- system_status = 7 chỉ ra rằng hệ thống đã bắt đầu quá trình tắt nguồn và hiện đang tắt.
- system_status = 8 chỉ ra rằng hệ thống tự kết thúc và kết thúc chuyến bay.

Cuối cùng, trường mavlink_version chỉ ra phiên bản MAVLink. Nó không thể chỉnh sửa bởi người dùng và được thiết lập bởi giao thức.

System Status message: thông báo trạng thái hệ thống có ID tin nhắn bằng 1 và bao gồm dữ liệu về các cảm biến điều khiển trên bo mạch được nhúng vào hệ thống không người lái và chỉ định những cảm biến nào được bật/tắt và cảm biến nào đang hoạt động hoặc có lỗi. Nó cũng thực hiện dữ liệu về trạng thái pin và điện áp còn lại, rất hữu ích để theo dõi mức pin của hệ thống không người lái. Bên cạnh đó, nó cung cấp thông tin về các lỗi giao tiếp và tỷ lệ các gói bị mất trong liên kết truyền thông. Thông tin về pin và liên kết truyền thông là rất quan trọng để thực hiện hành động không an toàn phù hợp khi mức pin giảm hoặc chất lượng liên lạc giảm. Trong trường hợp này, hệ thống không người lái có thể được lập trình sẵn để thực hiện một hoạt động không an toàn trong trường hợp mức pin thấp hoặc chất lượng liên lạc kém như hạ cánh và trở về nhà cho một hệ thống máy bay không người lái.

Global Position message: Thông báo định vị toàn cầu có id bằng 33 và thể hiện tọa độ GPS được lọc do cảm biến Định vị toàn cầu cung cấp. Nó được minh họa trong Hình 2.4. Thông báo này đưa ra thông tin quan trọng của hệ thống không người lái liên quan đến các vị trí toàn cầu của nó, cụ thể là vĩ độ GPS (lat), kinh độ (lon) và độ cao tuyệt đối (alt). Ba giá trị này được mã hóa thành bốn byte (32 bit). Giá trị của (lat) và (lon) phải được chia cho 107 , để có được giá trị GPS thực sự, cần phải chia chúng cho 107. Độ cao được biểu thị bằng milimét. Thông báo cũng chứa trường độ cao tương đối (Rel_alt), đại diện cho độ cao tương đối với điểm mặt đất cất cánh của hệ thống máy bay không người lái. Nó khác với độ cao tuyệt đối. Ví dụ, độ cao tuyệt đối của thành phố Riyadh là 612 mét, tương ứng với độ cao tương đối là 0 mét, vì máy bay không người lái đang ở trên mặt đất. Nếu máy bay không người lái cất cánh ở độ cao tương đối so với mặt đất 10 mét, thì độ cao tuyệt đối của nó trở thành 622 mét. Ngoài ra, thông báo này cũng cung cấp thông tin về tốc độ tuyến tính của hệ thống không người lái dọc theo trục 3 ((x, y, z) ngoài định hướng được gọi là tiêu đề (heading). Thông tin này được thu thập từ cảm biến GPS và cũng có thể được đọc từ các cảm biến khác như Đơn vị đo lường quán tính (IMU) hoặc la bàn, có sẵn trong các tin nhắn MAVLink khác.

time_boot_ms	Lat	Lon	alt	relative_alt	vx	vy	vz	Heading
32 bits	32 bits	32 bits	32 bits	32 bits	16 bits	16 bits	16 bits	16 bits

Hình 2.4 System status message

2.1.3.2 Thông báo lệnh

Có một số thông báo lệnh trong MAVLink cung cấp khả năng yêu cầu hệ thống không người lái thực hiện một số hành động nhất định. Em sẽ trình bày tổng quan về một số lệnh quan trọng nhất.

COMMAND LONG: COMMAND LONG là một lệnh đa mục đích cho phép gửi các loại lệnh khác nhau tùy thuộc vào loại lệnh của thông báo và các tham số của nó. Thông báo COMMAND LONG có Message ID (ID Thông báo) bằng 76 và được xác định bằng 11 trường, như được minh họa trong Hình 2.5. Trường target_system và trường target_component chỉ định hệ thống sẽ thực thi lệnh và thành phần bên dưới của nó.

Trường lệnh (The command field) đề cập đến loại lệnh sẽ được thực thi. Nó được định nghĩa trong bảng liệt kê lệnh MAV_CMD. Ngoài ra, đối với mỗi lệnh, một tập hợp các tham số liên quan đến lệnh có thể được đặt.

Ví dụ: ID lệnh với 21 đề cập đến lệnh Land và nó không có tham số. Một số lệnh khác có các tham số như lệnh cất cánh (ID = 22), phải chỉ định độ cao cất cánh trong param 7 và nhánh lệnh / giải giáp chỉ định giá trị Boolean trong param1 để cho biết có nên kích hoạt hoặc giải giáp động cơ hay không. Có khoảng 60 loại lệnh được định nghĩa trong bảng liệt kê lệnh MAV CMD. Trường xác nhận cho biết nếu tin nhắn được gửi lần đầu tiên với giá trị 0 và các giá trị khác thể hiện xác nhận tin nhắn. 7 tham số phụ thuộc vào loại lệnh. Ví dụ, đối với lệnh Land, tất cả bảy tham số đều vô dụng. Trong lệnh cất cánh, tham số thứ bảy thể hiện độ cao được yêu cầu khi cất cánh.

The Mission Item Command: Lệnh nhiệm vụ có ID Thông báo bằng 39 và cho phép gửi một điểm tham chiếu đến một hệ thống không người lái để nó điều hướng tự động đến điểm tham chiếu cụ thể đó trong chế độ GUIDED. Mỗi thông báo lệnh nhiệm vụ có một số thứ tự chỉ định số của nó trong nhiệm vụ, bắt đầu từ 0, chỉ định vị trí nhà. Nó cũng có ba trường (x, y, z), xác định tọa độ của điểm tham chiếu. Tuy nhiên, tọa độ phải được chỉ định liên quan đến khung tham chiếu. Do đó, thông báo có một trường được gọi là khung, trong đó chỉ định khung tọa độ tham chiếu của điểm tham chiếu. Tham số này rất quan trọng vì nó là điều cần thiết để giải thích ý nghĩa của tọa độ (x, y, z). Ví dụ: nếu chúng ta đặt tọa độ điểm tham chiếu là (24.68773, 46.72185, 10) với tham chiếu đến khung toàn cầu MAV FRAME GLOBAL, điều này có nghĩa là đi đến điểm tham chiếu tại vị trí GPS (24.68773, 46.72185) và độ cao tuyệt đối là 10 mét đối với mực nước biển. Ví dụ, tại thành phố Riyadh (Ả Rập Xê Út), độ cao tuyệt đối là 620 m, vì vậy điều này sẽ khiến máy bay không người lái có thể rơi xuống đất vì độ cao mục

tiêu thấp hơn mặt đất. Tuy nhiên, thông thường, muốn chỉ định vị trí đối với mặt đất (ví dụ: 10 mét so với mặt đất) và đối với điều này, ta cần chỉ định khung tham chiếu là khung toàn cầu nhưng với độ cao tương đối cụ thể là MAV FRAME GLOBAL Relocate ALT. Lệnh cũng chỉ định hệ thống đích và thành phần đích như trong các thông báo lệnh khác.

target system unit8_t	target component unit8_t	command unit16_t	confirmation unit8_t	param1 float	param2 float	param3 float	param4 float	param5 float	param6 float	param7 float

Hình 2.5 Command long

2.1.4 Chế độ bay

Hiểu các chế độ bay là vô cùng quan trọng để có thể điều khiển một hệ thống không người lái chạy Ardupilot và giao thức MAVLink. Có một số chế độ máy bay được xác định bởi Ardupilot. Trong phần này, em xin trình bày các chế độ bay quan trọng nhất và phổ biến hơn.

- **Chế độ STABILIZE**: trong chế độ này, hệ thống không người lái được điều khiển hoàn toàn bởi người dùng, vị trí, độ cao. Hệ thống không người lái sẽ phản hồi mọi đầu vào từ bộ điều khiển RC do người dùng điều khiển và tùy thuộc vào người dùng để bù bất kỳ sự trôi dạt nào do hệ thống không người lái thực hiện.
  Người dùng nên chuyển ngay sang chế độ STABILIZE để điều khiển thủ công hệ thống không người lái, khi chế độ lái tự động không điều khiển được xe ở bất kỳ chế độ tự trị nào khác.
  Cần lưu ý rằng có thể tải xuống tệp nhật ký dữ liệu từ hệ thống không người lái, để phân tích hiệu suất chuyến bay bằng cách mở nó trong trình lập kế hoạch nhiệm vụ trong tab biểu đồ.
- **ALTITUDE HOLD**: chế độ thoải mái hơn để điều khiển hệ thống không người lái là chế độ ALTITUDE HOLD hoặc ALT HOLD mà người dùng không phải lo lắng về việc duy trì độ cao cố định cho hệ thống không người lái, vì chế độ lái tự động sẽ đảm nhiệm việc điều khiển tự động bằng cách sử dụng PID điều khiển độ cao. Người dùng sẽ phải điều khiển hướng và vị trí của hệ thống không người lái bằng tay.
  Chế độ ALT_HOLD tự động kiểm soát độ cao của hệ thống không người lái bằng chế độ lái tự động. Tuy nhiên, nó không kiểm soát tiêu đề và vị trí còn lại cho người dùng. Chế độ này được khuyến nghị cho người mới hơn chế độ STABILIZE và không yêu cầu GPS vì nó ước tính độ cao với the barometer.
- **LOITER**: một chế độ thậm chí dễ truy cập hơn để điều khiển hệ thống không người lái là chế độ LOITER, duy trì vị trí, hướng và độ cao hiện tại của hệ thống không người lái một khi người dùng không cung cấp đầu vào cho bộ điều khiển RC.

- **LAND**: chế độ LAND sẽ buộc hệ thống không người lái hạ cánh xuống đất.
- **RTL**: chế độ RTL, còn được gọi là Return to Launch, sẽ buộc hệ thống không người lái quay trở lại vị trí bắt đầu nơi nó thực hiện cất cánh.
  Chế độ LAND và RTL được sử dụng trong trường hợp vi phạm an toàn điều hướng và địa chất, ví dụ, có thể lập trình trên chế độ lái tự động rằng nếu pin ở dưới một mức nhất định, thì hệ thống không người lái cần phải hạ cánh ngay lập tức hoặc quay trở lại vị trí bắt đầu tự động. Điều này được gọi là GEOFENCE.
- **GUIDED**: chế độ GUIDED rất cần thiết và chỉ hoạt động với chế độ GPS. Khi GPS của hệ thống không người lái thực hiện sửa lỗi 3D và được kích hoạt, thì hệ thống không người lái có thể được gửi để điều hướng tự động đến tọa độ GPS cụ thể thông qua trạm mặt đất. Nó được gọi là chế độ GUIDED vì hệ thống không người lái được hướng dẫn bởi người dùng để điều hướng tự động đến một điểm tham chiếu cụ thể do người dùng chọn.
  Chế độ GUIDED được sử dụng cùng với GPS và cho phép người dùng gửi hệ thống không người lái đến điểm tham chiếu được chỉ định theo tọa độ GPS của họ. Không thể kích hoạt hệ thống không người lái ở chế độ GUIDED chỉ khi GPS có sửa lỗi 3D.
  Trong chế độ GUIDED thường, một trạm mặt đất được sử dụng để gửi điểm tham chiếu điều hướng đến hệ thống không người lái để điều hướng đến nó. Do đó, điều quan trọng là phải có một thiết bị telemetry được kết nối với hệ thống không người lái và trạm mặt đất để cho phép liên lạc tầm xa giữa chúng. Thông thường, người dùng cần nhấp vào một điểm trên bản đồ, trong chế độ GUIDED và hệ thống không người lái sẽ lập kế hoạch đường đi của nó và tiến tới mục tiêu.
- AUTO: chế độ AUTO đề cập đến chế độ tự trị trong đó hệ thống không người lái sẽ thực hiện theo nhiệm vụ được xác định trước. Nhiệm vụ là một tập hợp các điểm mốc được lưu trữ trong hệ thống lái tự động không người lái. Khi chế độ AUTO được chọn, hệ thống không người lái sẽ tự động đi đến từng điểm tham chiếu theo cùng thứ tự khi chúng được lưu trữ.

2.2 Thư viện MAVSDK [3]

2.2.1 Tổng quan về thư viện MAVSDK

MAVSDK là Thư viện MAVLink với các API cho C ++, iOS, Python và Android.

Thư viện cung cấp một API đơn giản để quản lý một hoặc nhiều phương tiện, cung cấp quyền truy cập theo chương trình vào thông tin và từ xa của phương tiện, và kiểm soát các nhiệm vụ, di chuyển và các hoạt động khác.

Thư viện có thể chạy trên máy tính đồng hành trên xe hoặc trên GCS hoặc thiết bị di động trên mặt đất (những thiết bị này có sức mạnh xử lý đáng kể hơn một bộ điều khiển

bay thông thường, cho phép thực hiện các nhiệm vụ như tầm nhìn máy tính, tránh chướng ngại vật và lập kế hoạch tuyến đường).

Các nhà phát triển có thể mở rộng SDK C ++ lõi bằng cách sử dụng các plugin để thêm bất kỳ API MAVLink cần thiết nào khác (ví dụ: để tích hợp bộ điều khiển chuyến bay với máy ảnh tùy chỉnh, gimbals hoặc phần cứng khác trên MAVLink).

2.2.2 Các tính năng của thư viện MAVSDK

Thư viện lõi được viết bằng C ++, với các ràng buộc được tạo tự động cho các ngôn ngữ lập trình được hỗ trợ khác. Thư viện MAVSDK có các đặc điểm sau:

- Đơn giản và dễ sử dụng. Nó có API hỗ trợ cả cuộc gọi đồng bộ (chặn) và cuộc gọi không đồng bộ (sử dụng callback).
- Nhanh, mạnh mẽ và nhẹ. Được xây dựng để xử lý việc sử dụng trên tàu với tin nhắn tốc độ cao.
- Đa nền tảng (Linux, macOS, Windows, iOS, Android).
- Mở rộng, sử dụng các plugin thời gian biên dịch.

Các tính năng chính được cung cấp bởi API lõi (trong tất cả các ngôn ngữ lập trình):

- Kết nối và quản lý tối đa 255 phương tiện thông qua kết nối TCP, UDP hoặc nối tiếp.
- Nhận thông tin về xe cộ (nhà cung cấp, phiên bản phần mềm, phiên bản sản phẩm, v.v.)
- Nhận telemetry phương tiện và thông tin trạng thái (ví dụ: ắc quy, GPS, kết nối RC, chế độ máy bay, v.v.) và đặt tốc độ cập nhật từ xa.
- Gửi lệnh sẵn sàng, hủy sẵn sàng, kill, cất cánh, hạ cánh và trở về điểm cất cánh.
- Tạo và quản lý các nhiệm vụ.
- Điều khiển một camera và gimbal cả bên trong và bên ngoài nhiệm vụ.
- Gửi lệnh để trực tiếp điều khiển chuyển động của xe.
- Gửi lệnh để bắt đầu hiệu chuẩn cảm biến.

2.2.3 Thư viện MAVSDK Core C++

MAVSDK Core cung cấp API C ++ đơn giản để quản lý một hoặc nhiều phương tiện thông qua MAVLink. Nó cho phép truy cập theo chương trình vào thông tin xe và telemetry, và kiểm soát các nhiệm vụ, chuyển động và các hoạt động khác. Thư viện C ++ rất hiệu quả và có thể được sử dụng để cho phép các tác vụ như thị giác máy tính, tránh chướng ngại vật và lập kế hoạch tuyến đường.

Các nhà phát triển có thể mở rộng SDK, sử dụng các plugin để thêm bất kỳ API MAVLink cần thiết nào khác (ví dụ: để tích hợp ngăn xếp chuyến bay với máy ảnh tùy chỉnh, gimbals hoặc phần cứng khác trên MAVLink).

Thư viện lõi được sử dụng để cung cấp triển khai cơ bản của các thư viện MAVSDK khác - ví dụ: MAVSDK-Swift.

2.2.3.1 Tổng quan về API

Mavsdk là lớp thư viện chính. Người sử dụng API dùng Mavsdk để khám phá và truy cập phương tiện (Đối tượng hệ thống), từ đó cung cấp quyền truy cập vào tất cả các đối tượng điều khiển và thông tin không người lái khác (ví dụ: Telemetry, Mission, v.v.).

Các lớp quan trọng nhất là:

- **Mavsdk:** khám phá và kết nối với các phương tiện (Hệ thống).
- **System:** đại diện cho một phương tiện được kết nối (ví dụ: máy bay không người lái hoặc máy bay không người lái VTOL). Nó cung cấp quyền truy cập vào thông tin xe và kiểm soát thông qua các lớp được liệt kê dưới đây.
- **Info:** thông tin phiên bản cơ bản về phần cứng và phần mềm của hệ thống.
- **Telemetry:** nhận thông tin telemetry và thông tin trạng thái của xe và thiết lập tốc độ cập nhật telemetry.
- **Action:** các hành động bay không người lái đơn giản bao gồm sẵn sàng, cất cánh và hạ cánh.
- **Mission:** tạo nhiệm vụ Waypoint và tải lên / tải xuống. Nhiệm vụ được tạo ra từ các đối tượng MissionItem.
- **Offboard:** điều khiển máy bay không người lái bằng các lệnh vận tốc.
- **Geofence:** chỉ định một geofence.
- **Gimbal:** kiểm soát một gimbal.
- **Camera:** điều khiển camera.
- **FollowMe:** drone theo dõi một vị trí được cung cấp bởi SDK.
- **Calibration:** hiệu chỉnh cảm biến (ví dụ: con quay hồi chuyển, gia tốc kế và từ kế).
- **LogFiles:** tải tập tin nhật ký từ xe.

Các API sau cung cấp quyền truy cập trực tiếp hơn vào các loại thông điệp MAVLink cơ bản. Chúng chỉ nên được sử dụng khi các tính năng bị thiếu trong các API chính ở trên.

- **Param**: truy cập thô để lấy và đặt tham số.
- **MissionRaw**: truy cập trực tiếp vào các mục nhiệm vụ MAVLink.
- **MavlinkPassthrough**: cung cấp quyền truy cập MAVLink đầy đủ / trực tiếp.

2.2.3.2 Cách sử dụng MAVSDK

Quản lý đối tượng: Mavsdk là lớp thư viện chính. Các ứng dụng phải tạo một đối tượng Mavsdk và phá hủy nó trong khi tắt ứng dụng. Đối tượng thường được tạo như

một biến tự động được dọn sạch khi đi ra khỏi phạm vi (ta cũng có thể tự động tạo/hủy đối tượng bằng cách sử dụng *new/delete*).

Người sử dụng API dùng Mavsdk để khám phá và kết nối với các đối tượng Hệ thống (phương tiện / máy ảnh, v.v.).

Truy cập vào thông tin máy bay không người lái và các đối tượng điều khiển được cung cấp bởi các plugin (ví dụ: Từ xa, Hành động, Nhiệm vụ, v.v.). Các đối tượng plugin được khởi tạo với một đối tượng hệ thống cụ thể (một phiên bản plugin phải được tạo cho mọi hệ thống cần nó).

Các đối tượng plugin được tạo như các con trỏ chia sẻ sẽ bị hủy khi tất cả các thẻ điều khiển liên quan nằm ngoài phạm vi. Tất cả các đối tượng được tự động dọn sạch khi đối tượng Mavsdk gốc bị hủy.

Xử lý lỗi: API MAVSDK không đưa ra ngoại lệ. Thay vào đó, các phương thức có thể không trả về thành công hoặc lỗi là giá trị enum.

Mã lỗi thường phản ánh sự thừa nhận từ chiếc xe rằng nó sẽ thực hiện hành động được yêu cầu (hoặc không). Bản thân thao tác có thể chưa hoàn thành (ví dụ: cất cánh).

Điều khiển phương tiện chia sẻ: một phương tiện có thể nhận lệnh từ nhiều nguồn, bao gồm trạm điều khiển mặt đất hoặc các ứng dụng MAVLink khác.

Các ứng dụng SDK, đang chạy trong môi trường phù hợp, có thể giám sát rõ ràng các thay đổi trong chế độ máy bay (kiểm soát ứng dụng bên ngoài) và thay đổi hành vi phù hợp.

2.2.3.3 Giới hạn của thư viện MAVSDK

Phương tiện được hỗ trợ: SDK đã được thiết kế để quản lý các máy bay sử dụng hệ thống lái tự động PX4. Nó chủ yếu đã được thử nghiệm để sử dụng với máy bay trực thăng, nhưng cũng có hỗ trợ cơ bản cho cánh cố định và VTOL.

API bao gồm các phương pháp không có ý nghĩa đối với các loại phương tiện khác - bao gồm "cất cánh" và "hạ cánh". Trong khi phương tiện mặt đất có thể hoạt động, chúng không được hỗ trợ và chưa được kiểm tra. Tương tự, các autopilots khác có thể hoạt động tốt, nhưng chúng không được hỗ trợ rõ ràng và chưa được kiểm tra.

Telemetry/Thông tin: SDK nhận và lưu trữ thông tin trạng thái/telemetry của xe từ các tin nhắn MAVLink nhận được. Thông tin được cung cấp cho các đăng kí callback ngay khi nhận được cập nhật tin nhắn. Người sử dụng cũng có thể truy vấn API một cách đồng bộ và sẽ nhận được thông tin từ tin nhắn nhận được cuối cùng (tùy thuộc vào độ trễ của kênh, thông tin này sẽ ngày càng trở nên "cũ" giữa các tin nhắn).

Thông tin thông tin không thay đổi cho một chiếc xe cụ thể, vì vậy sẽ chính xác bất cứ khi nào đọc.

Hành động/Offboard: phương thức hành động trả lại giá trị khi phương tiện đã xác nhận rằng tin nhắn đã được nhận và sẽ được thực hiện (hoặc không). Các phương thức không chờ đợi hành động được hoàn thành.

Vì vậy, ví dụ, phương thức Action :: land () trả lại giá trị ngay khi phương tiện xác nhận nó sẽ hạ cánh, nhưng thực tế sẽ hạ cánh tại một thời gian sau đó.

Các nhà phát triển sẽ cần giám sát riêng việc hoàn thành các hành động được yêu cầu, nếu điều này quan trọng đối với ứng dụng.

Nhiệm vụ: API Mission và MissionItem cung cấp một tập hợp con hữu ích nhất của các lệnh nhiệm vụ MAVLink dưới dạng API thân thiện với nhà phát triển.

Không phải mọi hành vi lệnh nhiệm vụ được giao thức hỗ trợ và PX4 sẽ được SDK hỗ trợ. Ví dụ: tại thời điểm viết API không cho phép chúng ta chỉ định các lệnh nhảy trở lại các mục nhiệm vụ trước đó.

API cho phép ta tải xuống/nhập các nhiệm vụ. Tuy nhiên, điều này sẽ thất bại nếu tác vụ chứa lệnh không được API hỗ trợ.

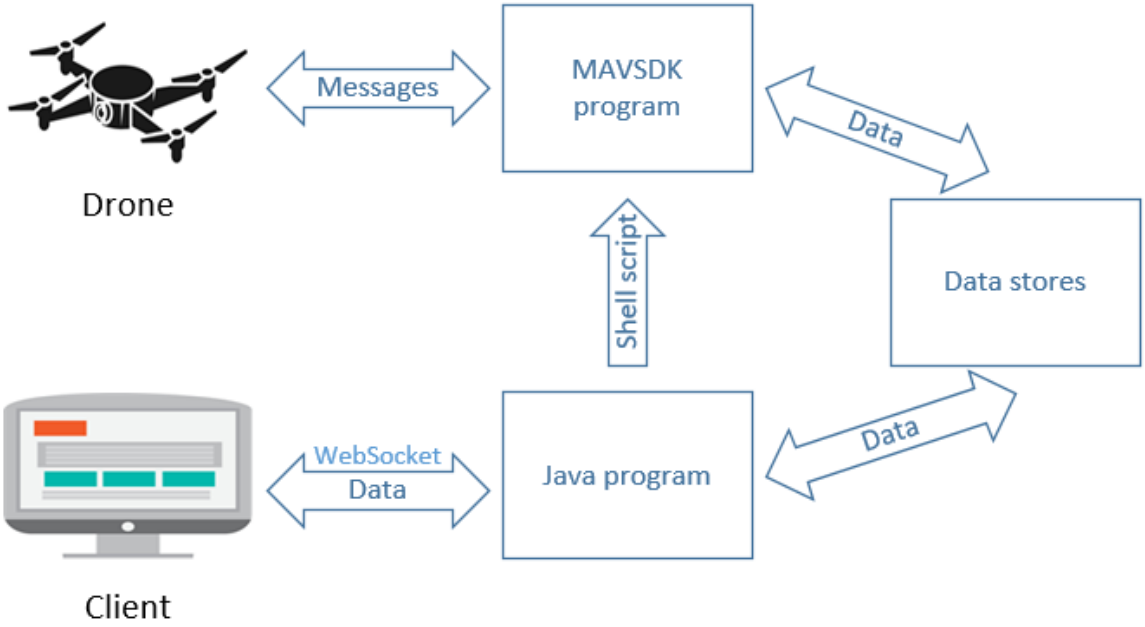
Tình trạng kết nối (connection status): một hệ thống được coi là bị ngắt kết nối (time-out) nếu heartbeat message của nó không được phát hiện trong vòng 3 giây.

Như vậy ở trong chương này, em đã trình bày một cách tổng quan về giao thức liên lạc MAVLink: về định nghĩa, các phiên bản, cấu trúc và các loại tin nhắn MAVLink. Bên cạnh đó, em cũng trình bày tổng quan về thư viện MAVSDK, một thư viện rất mạnh mẽ và hỗ trợ rất nhiều ngôn ngữ được phát triển từ MAVLink. MAVSDK cũng chính là thư viện mà em sẽ sử dụng để viết các chương trình giao tiếp và điều khiển drone. Ở chương tiếp theo, em sẽ xây dựng một phần mềm (trạm mặt đất) giao tiếp và điều khiển drone đơn giản.

CHƯƠNG 3. XÂY DỰNG TRẠM MẶT ĐẤT GIAO TIẾP VÀ QUẢN LÝ THIẾT BỊ BAY

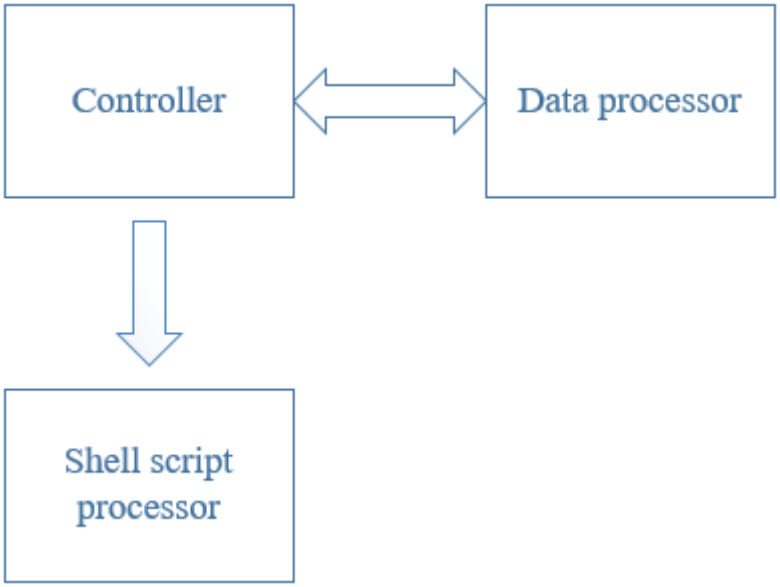
Ở chương này, em sẽ vận dụng các kiến thức đã tìm hiểu được để xây dựng một phần mềm dùng để giao tiếp và quản lý thiết bị bay trên môi trường Ubuntu.

3.1 Sơ đồ hệ thống phần mềm giao tiếp và quản lý thiết bị bay



Hình 3.1 Sơ đồ hoạt động phần mềm

3.1.1 Sơ đồ hoạt động Java program



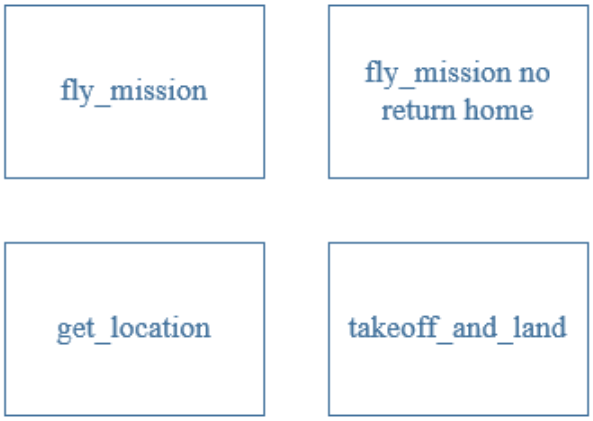
Hình 3.2 Sơ đồ hoạt động Java program

Controller: là nơi để tiếp nhận các request và trả về các response với Client qua WebSocket. Với mỗi lời gọi tới từ client, controller sẽ có những phương thức xử lý riêng, sau đó gọi tới Data processor hoặc Shell script processor tùy theo yêu cầu.

Data processor: nơi đọc/ghi và xử lý dữ liệu. Data processor tương tác hai chiều với Controller và file lưu trữ dữ liệu.

Shell script processor: nơi xử lý việc gọi chương trình MAVSDK thông qua lệnh hệ thống. Tương tác giữa Controller và Shell script processor là một chiều.

3.1.2 Sơ đồ hoạt động MAVSDK program



Hình 3.3 Các chương trình của MAVSDK

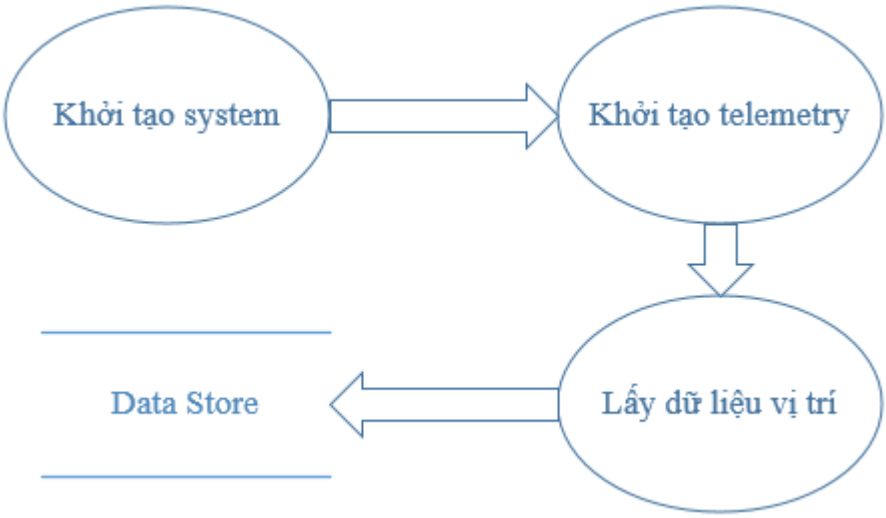
fly_mission: chương trình điều khiển drone bay theo nhiệm vụ do người dùng thiết lập, sau khi bay xong drone sẽ quay trở về vị trí xuất phát và hạ cánh

fly_mission no return home: chương trình điều khiển drone bay theo nhiệm vụ do người dùng thiết lập, sau khi bay xong drone sẽ hạ cánh ngay tại điểm cuối cùng của nhiệm vụ.

get_location: chương trình lấy về vị trí hiện tại của drone, dùng khi khởi động phần mềm hoặc khi kết thúc nhiệm vụ.

takeoff_and_land: chương trình điều khiển drone cất cánh và hạ cánh trong vòng 15 giây.

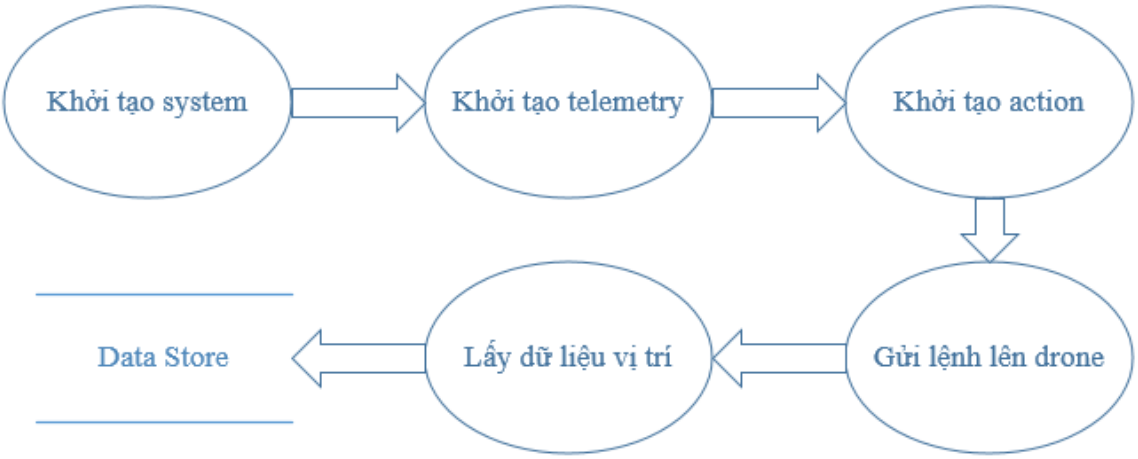
3.1.2.1 Chu trình hoạt động của chương trình get_location



Hình 3.4 Chu trình hoạt động của chương trình get_location

Đầu tiên, chương trình sẽ đăng kí theo dõi một cổng giao tiếp để SDK giám sát cho các hệ thống mới. Sau đó, SDK sẽ tự động phát hiện các hệ thống được kết nối. Khi phát hiện ra hệ thống mới, chương trình sẽ tạo đối tượng telemetry để lấy về thông tin được gửi từ drone. Ở trong chương trình get_location, ta chỉ lấy về vị trí hiện tại của drone, ghi vào data store sau đó thoát khỏi chương trình và ngắt kết nối với drone.

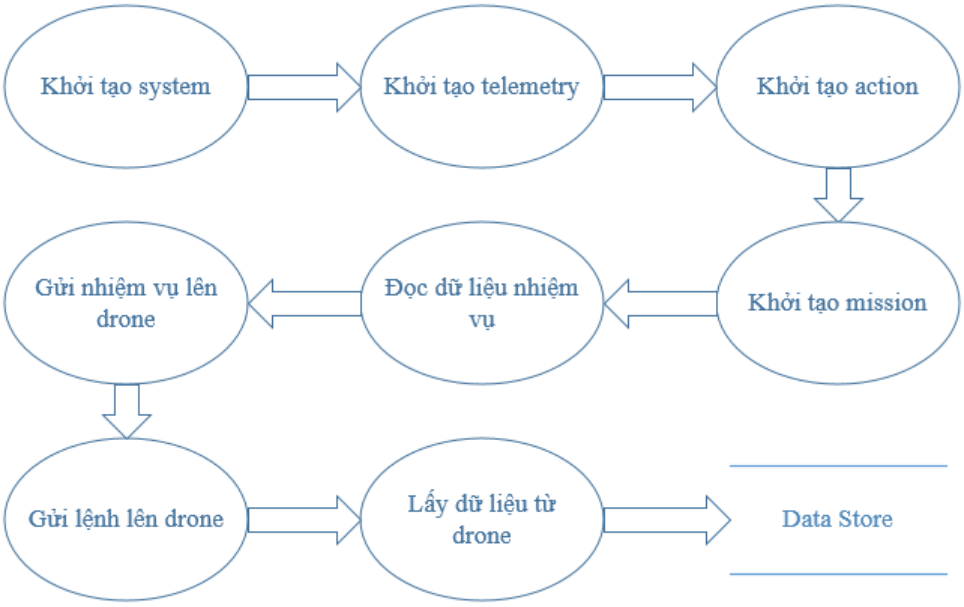
3.1.2.2 Chu trình hoạt động của chương trình takeoff_and_land



Hình 3.5 Chu trình hoạt động của chương trình takeoff_and_land

Đầu tiên, chương trình sẽ đăng kí theo dõi một cổng giao tiếp để SDK giám sát cho các hệ thống mới. Sau đó, SDK sẽ tự động phát hiện các hệ thống được kết nối. Khi phát hiện ra hệ thống mới, chương trình sẽ tạo đối tượng telemetry để lấy về thông tin được gửi từ drone và đối tượng action để gửi lệnh tới drone. Sau khi đối tượng action gửi lệnh tới drone, đối tượng telemetry sẽ theo dõi và lấy về vị trí của drone mỗi 2.5 giây và ghi vào data store. Sau khi drone kết thúc hành động và hạ cánh xuống mặt đất, chương trình sẽ kết thúc và ngắt kết nối với drone.

3.1.2.3 Chu trình hoạt động của fly_mission và fly_mission no return home



Hình 3.6 Chu trình hoạt động của chương trình fly_mission

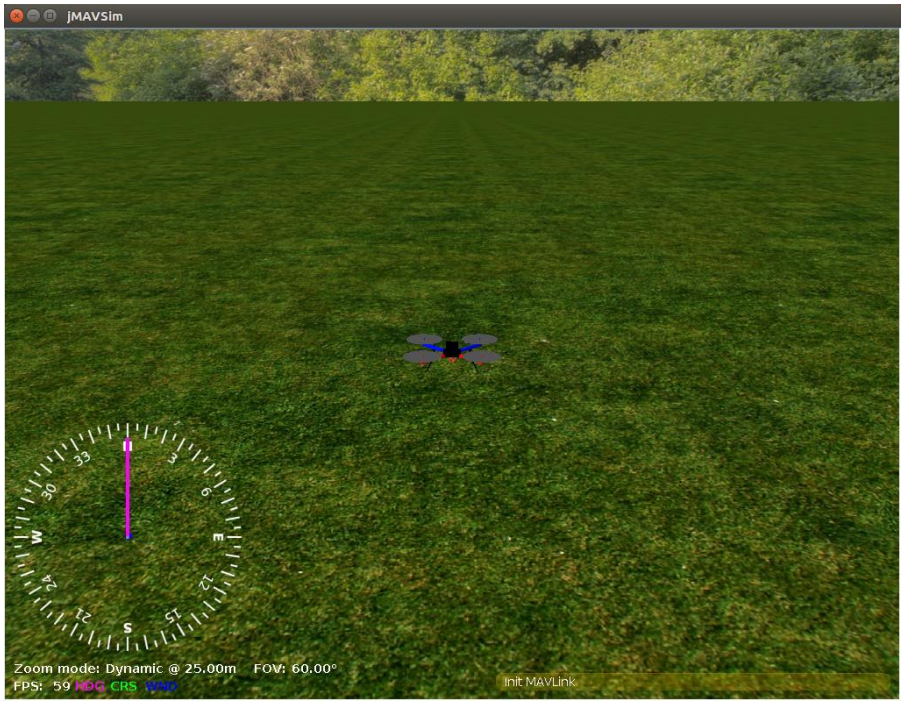
Đầu tiên, chương trình sẽ đăng kí theo dõi một cổng giao tiếp để SDK giám sát cho các hệ thống mới. Sau đó, SDK sẽ tự động phát hiện các hệ thống được kết nối. Khi phát hiện ra hệ thống mới, chương trình sẽ tạo đối tượng telemetry, action và đối tượng mission để cài đặt nhiệm vụ cho drone. Chương trình sẽ đọc nhiệm vụ từ data store, sau đó khởi tạo các nhiệm vụ và gửi chúng lên drone. Sau khi đối tượng action gửi lệnh tới

drone, đối tượng telemetry sẽ theo dõi và lấy về vị trí của drone mỗi 4 giây và ghi vào data store. Sau khi drone kết thúc nhiệm vụ, đối với chương trình fly_mission, action sẽ gửi lệnh quay về vị trí xuất phát, còn đối với chương trình fly_mission no return home, action gửi lệnh hạ cánh xuống mặt đất, chương trình sẽ kết thúc và ngắt kết nối với drone.

3.2 Xây dựng chi tiết các khối của phần mềm

3.2.1 Mô phỏng drone

Ở trong phạm vi của đồ án, em sử dụng phần mềm mô phỏng jMAVSim để mô phỏng drone. jMAVSim là một trình giả lập multirotor/Quad đơn giản cho phép ta lái các loại xe copter chạy PX4 quanh một thế giới giả lập. Ta có thể dễ dàng thiết lập và kiểm tra xem drone có thể cất cánh, bay, hạ cánh và phản ứng phù hợp với các điều kiện hỏng hóc khác nhau (ví dụ: lỗi GPS).



Hình 3.7 Giao diện mô phỏng jMAVSim

3.2.2 Khối lưu trữ dữ liệu

Dữ liệu: Dữ liệu ở đây có 2 loại, 1 là dữ liệu về vị trí (độ cao, kinh độ, vĩ độ) của drone, 2 là dữ liệu về độ cao, vận tốc bay khi cài đặt nhiệm vụ cho drone. Cấu trúc của data về vị trí khi được truyền ở WebSocket như sau:

```
{
  "alt": number,
  "lat": number,
  "lon": number
}
```

Trong đó: alt (altitude) là độ cao của drone; lat (latitude) là vĩ độ của drone, lon (longitude) là kinh độ của drone.

Cấu trúc của dữ liệu về độ cao và vận tốc bay khi được truyền ở WebSocket như sau:

```
{
  "strAltsp": string
}
```

Trong đó, strAltsp là chuỗi lưu trữ thông tin về độ cao và tốc độ của drone.

Data stores: Nơi lưu trữ dữ liệu từ client gửi lên và từ drone gửi về. Dữ liệu được lưu trữ dưới dạng file text, được chia thành các file: input, output, location, altsp chứa thông tin như sau:

- input: Chứa thông tin về tọa độ cho các nhiệm vụ của drone, kinh độ viết trước, vĩ độ viết sau, ngăn cách bằng dấu phẩy, tọa độ của các điểm nhiệm vụ sẽ ngăn cách nhau bằng kí tự xuống dòng "\n":

```
8.548646382942216,47.399051931762024
8.548656382942216,47.399151931762024
8.548666382942216,47.399251931762024
```

- output: Chứa thông tin về tọa độ được trả về từ drone (dùng trong trường hợp theo dõi drone khi thực hiện cất cánh/hạ cánh và thực hiện nhiệm vụ), độ cao viết trước, vĩ độ và kinh độ viết sau, ngăn cách nhau bằng dấu phẩy, mỗi tọa độ trả về sẽ được ngăn cách nhau bằng kí tự xuống dòng "\n":

```
0.455,47.3985,8.54789
6.847,47.3985,8.54789
15.287,47.3985,8.54789
16.635,47.3985,8.54794
16.746,47.3987,8.54812
```

- location: Chứa thông tin về tọa độ hiện tại của drone, thường dùng khi mới bắt đầu khởi động phần mềm hoặc khi thực hiện xong nhiệm vụ nhưng không trở về nơi cất cánh ban đầu, có cấu trúc như input, nhưng thay đổi vị trí kinh độ và vĩ độ cho nhau.
- altsp: Chứa thông tin về độ cao và tốc độ được cài đặt khi drone thực hiện nhiệm vụ. Độ cao viết trước, tốc độ viết sau, ngăn cách nhau bằng dấu phẩy:

```
17,8
```

3.2.3 Chương trình MAVSDK

Các chương trình MAVSDK dùng để điều khiển drone được viết bằng C++, sử dụng thư viện MAVSDK với các API cho C++, sử dụng công cụ Cmake để xây dựng chương trình.

Để có thể viết được các chương trình sử dụng MAVSDK, ta phải xây dựng và cài đặt thư viện MAVSDK từ source code của nó. Cách xây dựng và cài đặt thư viện MAVSDK có thể tham khảo tại [4].

Có 4 chương trình điều khiển được xây dựng với 4 mục đích khác nhau:

- get_location program: chương trình lấy về vị trí hiện tại của drone, thường được dùng khi bắt đầu khởi động phần mềm hoặc khi drone kết thúc 1 nhiệm vụ mà không quay lại điểm cất cánh.
- takeoff_and_land: chương trình ra lệnh cho drone cất cánh và hạ cánh trong thời gian 15 giây, vị trí của drone sẽ được cập nhật mỗi 2.5 giây.
- fly_mission: chương trình ra lệnh cho drone thực hiện nhiệm vụ, bay tới các vị trí được người dùng chỉ định, sau đó quay lại điểm cất cánh ban đầu, với độ cao và tốc độ sau mỗi vị trí do người dùng nhập hoặc mặc định sẽ là 10m và 5m/s, vị trí của drone sẽ được cập nhật mỗi 4 giây.
- fly_mission_norl: giống như với chương trình fly_mission, nhưng sau khi kết thúc nhiệm vụ, nó sẽ không trở lại vị trí cất cánh mà sẽ hạ cánh ngay tại điểm cuối cùng của nhiệm vụ.

Thư viện MAVSDK cung cấp rất nhiều API để ta có thể viết các hàm xử lý khác nhau. Tuy nhiêm trong phạm vi của đồ án, em sẽ trình bày các API quan trọng nhất và cách sử dụng chúng để xây dựng phần mềm (mã nguồn được trình bày trong phần phụ lục):

Kết nối với hệ thống:

Để phát hiện các phương tiện, trước tiên ta phải chỉ định các cổng giao tiếp mà SDK sẽ giám sát cho các hệ thống mới. Sau khi giám sát một cổng, SDK sẽ tự động phát hiện các phương tiện được kết nối, thêm chúng vào danh sách của nó và thông báo cho người dùng đã đăng ký về các sự kiện kết nối và ngắt kết nối.

- **Giám sát một cổng**: chỉ định một hoặc nhiều cổng để theo dõi bằng một trong các phương thức kết nối (đồng bộ): add_any_connection(), add_udp_connection(), add_tcp_connection() hoặc add_serial_connection(). Tất cả các phương thức được sử dụng tương tự và trả về ngay lập tức với ConnectionResult cho biết liệu chúng có thành công hay không.
  Phương thức add_any_connection() có thể được sử dụng để thiết lập giám sát cho bất kỳ loại cổng được hỗ trợ nào (trong khi các phương thức khác thiết lập các

loại kết nối cụ thể). Trong chương trình này em sử dụng phương thức add_any_connection().

- **Đăng ký thông báo phát hiện hệ thống**: SDK giám sát mọi cổng giao tiếp được thêm cho các hệ thống mới, được phân biệt bằng UUID xe. Ta có thể đăng ký thông báo khi hệ thống mới được phát hiện bằng register_on_discover() và cho hệ thống hết thời gian (không còn kết nối) bằng register_on_timeout().
  Các phương thức được sử dụng theo cùng một cách và gọi hàm callback với UUID của hệ thống được phát hiện/ngắt kết nối. UUID này sau đó có thể được sử dụng để lấy một đối tượng hệ thống để quản lý phương tiện liên quan.
  Nếu một hệ thống không có UUID thì Mavsdk thay vào đó sẽ sử dụng ID hệ thống MAVLink của nó (một số trong phạm vi từ 1 đến 255). Trên mạng MAVLink được cấu hình đúng, điều này sẽ là duy nhất.
- **Truy cập hệ thống**: khi Mavsdk đã cung cấp cho ta một UUID phương tiện, ta có thể sử dụng phương thức Mavsdk::system() để lấy đối tượng hệ thống liên quan.
  Hệ thống được sử dụng bởi các lớp plugin MAVSDK để truy vấn và điều khiển phương tiện.

Telemetry:

Lớp Telemetry được sử dụng để lấy từ xa phương tiện, bao gồm thông tin về trạng thái và chế độ máy bay.

Tất cả các hàm đều có cả phiên bản đồng bộ và không đồng bộ và ta có thể đặt tốc độ mà phương tiện cung cấp cập nhật cho từng loại thông tin. Tất cả các phương thức của một loại cụ thể (phương thức đồng bộ, không đồng bộ và set_rate) được sử dụng theo cùng một cách.

- **Tạo plugin Telemetry**: liên kết thư viện plugin vào ứng dụng. Thực hiện việc này bằng cách thêm mavsdk_telemetry vào phần target_link_lologists của tệp định nghĩa xây dựng cmake của ứng dụng.
  Tạo kết nối đến một system.
  Tạo một con trỏ được chia sẻ với một thể hiện của Telemetry được khởi tạo với system.
  Con trỏ Telemetry sau đó có thể được sử dụng để truy cập API plugin.
- **Đặt tỷ lệ cập nhật**: tốc độ cập nhật Telemetry xác định tần suất callback sẽ được gọi với thông tin mới và cũng là "độ mới" của dữ liệu thu được khi sử dụng API từ xa đồng bộ. Tốc độ cập nhật mặc định phụ thuộc vào chế độ lái tự động và cũng có thể bị giới hạn bởi các đặc điểm của kênh liên lạc. Ta có thể đặt tốc độ cho từng loại từ xa và cả hai phương pháp cài đặt tốc độ đồng bộ hoặc không đồng bộ đều được cung cấp.

- **Chặn yêu cầu Telemetry (State Management)**: một số lệnh/thao tác xe phải được thực hiện theo một trình tự nhất định hoặc chỉ có thể được gọi khi xe đạt đến một trạng thái nhất định. Ví dụ, để cất cánh, trước tiên xe phải được sẵn sàng, và để thực hiện sẵn sàng, xe phải có vị trí ban đầu/khóa GPS. Đối với những trường hợp này, ta sẽ theo dõi chiếc xe đang ở trạng thái nhất định và chặn các lệnh khác cho đến khi nó sẵn sàng để tiếp tục.

Actions (Take-off, Landing, Arming):

Lớp Action được sử dụng để điều khiển phương tiện sẵn sàng, cất cánh, hạ cánh, trở về vị trí ban đầu và hạ cánh, hũy sẵn sàng, kill và chuyển đổi giữa các chế độ VTOL. Các phương thức gửi lệnh đến một phương tiện và return/complete với phản ứng của phương tiện. Lưu ý rằng một phản hồi thành công cho biết liệu chiếc xe có ý định hành động theo lệnh hay không, chứ không phải là nó đã kết thúc hành động (ví dụ: vũ sẵn sàng, hạ cánh, cất cánh, v.v.).

- **Tạo Plugin Action**: liên kết thư viện plugin vào ứng dụng. Thực hiện việc này bằng cách thêm mavsdk_action vào phần target_link_library của tệp định nghĩa xây dựng cmake của ứng dụng.
  Tạo kết nối đến một system.
  Tạo một con trỏ được chia sẻ với một thể hiện của Action được khởi tạo với system.
  Con trỏ Action sau đó có thể được sử dụng để truy cập API plugin.
- **Takeoff (cất cánh)**: sử dụng các phương thức takeoff() hoặc takeoff_async(). Nếu lệnh cất cánh được chấp nhận, xe sẽ chuyển sang chế độ Takeoff, bay đến độ cao được cài đặt, sau đó giữ ở chế độ cất cánh cho đến khi nhận được lệnh khác.
  Drone sẽ chỉ cất cánh một khi được sẵn sàng, và chỉ có thể sẵn sàng một khi nó "khỏe mạnh" (đã được hiệu chỉnh, vị trí bắt đầu đã được đặt và có khóa GPS chất lượng đủ cao). Sau khi nó bắt đầu bay, mã cần kiểm tra việc cất cánh đã hoàn thành trước khi gửi hướng dẫn bổ sung.
- **Nhận/Cài đặt độ cao cất cánh**: độ cao cất cánh mặc định/hiện tại có thể được truy vấn bằng cách sử dụng get_takeoff_altitude(). Mục tiêu này có thể được thay đổi tại bất kỳ thời điểm nào trước khi cất cánh bằng cách sử dụng set_takeoff_altitude().
- **Landing (hạ cánh)**: cách tốt nhất để hạ cánh phương tiện tại vị trí hiện tại là sử dụng các phương thức land() hoặc land_async (). Nếu lệnh được chấp nhận, xe sẽ chuyển sang chế độ Land và hạ cánh tại điểm hiện tại.
- **Return/RTL**: chế độ return đưa phương tiện trở về vị trí bắt đầu và cũng có thể hạ cánh phương tiện (tùy theo cấu hình phương tiện). Chế độ này được gọi từ Action bằng cách sử dụng phương thức return_to_launch() hoặc

return_to_land_async(). Tùy thuộc vào drone, nó có thể hạ cánh hoặc bay xung quanh điểm trở về.

- **Disarm/Kill**: sử dụng các phương thức disarm hoặc kill để dừng động cơ máy bay không người lái (điểm khác biệt là việc giải giáp sẽ chỉ thành công nếu máy bay tự động hạ cánh, trong khi kill sẽ vô hiệu hóa một phương tiện ngay cả khi đang bay).

Nhiệm vụ (Missions):

API nhiệm vụ (plugin) cho phép ta tạo, tải lên, tải xuống, nhập từ QGroundControl, chạy, tạm dừng, khởi động lại, theo dõi các nhiệm vụ. Các nhiệm vụ có thể có nhiều "mission items", mỗi mission item có thể chỉ định vị trí, độ cao, hành vi bay qua, hành động của camera, vị trí gimbal và tốc độ sử dụng khi di chuyển đến vị trí tiếp theo.

- **Tạo Plugin Mission**: liên kết thư viện plugin vào ứng dụng. Thực hiện việc này bằng cách thêm mavsdk_mission vào phần target_link_libraries của tệp định nghĩa xây dựng cmake của ứng dụng.
  Tạo kết nối đến system.
  Tạo một con trỏ được chia sẻ với một thể hiện của Mission được khởi tạo với system.
  Con trỏ Mission sau đó có thể được sử dụng để truy cập API plugin.
- **Xác định một nhiệm vụ**: một nhiệm vụ phải được định nghĩa là một vectơ của các đối tượng MissionItem. Ta có thể tạo bao nhiêu đối tượng MissionItem tùy thích và sử dụng std_vector::push_back() để thêm chúng vào cuối của vectơ nhiệm vụ.
- **Tải lên một nhiệm vụ**: sử dụng Mission::upload_mission() để tải lên nhiệm vụ được xác định trong mục trên.
- **Bắt đầu/tạm dừng nhiệm vụ**: bắt đầu hoặc tiếp tục một nhiệm vụ bị tạm dừng bằng Mission::start_mission_async(). Drone phải có một nhiệm vụ (nhiệm vụ không cần phải được tải lên bằng SDK). Để tạm dừng một nhiệm vụ, ta sử dụng Mission::pause_mission_async().

Như vậy ở trong mục này em đã trình bày cách xây dựng chương trình từ thư viện MAVSDK, đây chính là phần cốt lõi của hệ thống giúp chúng ta có thể giao tiếp và quản lý drone.

3.2.4 Chương trình Java và giao diện người dùng

Để người dùng có thể dễ dàng thao tác với việc giao tiếp và quản lý drone, em quyết định chọn xây dựng một chương trình theo dõi và quản lý dựa trên nền tảng Web, cụ thể là dùng WebSocket để đảm bảo tính Realtime của hệ thống. Với giao diện theo dõi chính được xây dựng bằng Bing Map APIs. Nền tảng Bing Map cung cấp một bộ điều khiển và API dịch vụ mà ta có thể sử dụng để thêm Bing Map hoặc các dịch vụ không gian địa

lý vào ứng dụng của mình. Hơn nữa, Bing Map là hoàn toàn miễn phí. Trong phạm vi đồ án, Bing Map được sử dụng để tạo ra giao diện bản đồ, tạo ra các điểm theo dõi dựa trên tọa độ thu về từ drone, và chức năng thêm nhiệm vụ cho drone bằng sự kiện click chuột của người dùng trên bản đồ. Vấn đề đặt ra là: làm sao một Web có thể tác động vào drone trong khi chương trình để giao tiếp và quản lý drone được viết bằng C++?

Như ở phần MAVSDK Program em đã trình bày ở trên, các chương trình giao tiếp và quản lý drone được xây dựng dựa trên Cmake. Điều đó có ý nghĩa gì? Tức là chúng ta có thể gọi các chương trình này bằng shell script. Và như Hình 3.1, ta có thể thấy được, từ Java Program, ta có thể sử dụng các lệnh shell script để gọi tới MAVSDK Program. Đúng như thế, Java là một ngôn ngữ cực kì mạnh mẽ, nó chứa 1 class là ProcessBuilder. Trong Java, ta sẽ dùng ProcessBuilder để thực hiện gọi tới các chương trình bên ngoài, các lệnh của hệ điều hành.

Vấn đề móc nối giữa 2 chương trình đã được giải quyết, nhưng vậy còn dữ liệu để giao tiếp giữa drone và client thì sao? MAVSDK Program và Java Program sẽ không hề trao đổi dữ liệu trực tiếp với nhau, mà nó sẽ thông qua một phần trung gian chính là Data Stores em đã trình ở trên. Cả Java và C++ đều có thể thao tác đọc/ghi với các file text, việc cần giải quyết chính là làm sao để cho 2 chương trình sẽ không bị xung đột khi thực hiện đọc/ghi file.

Như vậy vấn đề về ngôn ngữ và dữ liệu đã được giải quyết, việc của chúng ta bây giờ đó là xây dựng luồng hoạt động của Web để người dùng có thể thao tác giao tiếp và quản lý drone một cách dễ dàng.

Web được xây dựng với một số chức năng chính như sau:

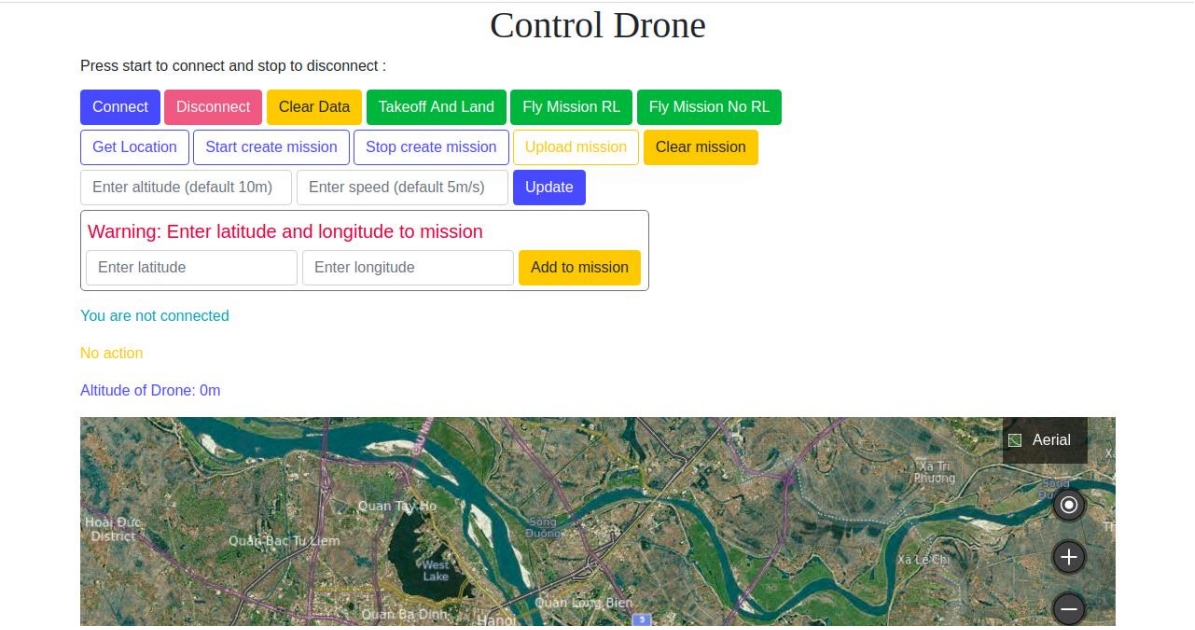
Lấy vị trí hiện tại của drone: được sử dụng khi bắt đầu khởi chạy ứng dụng hoặc khi kết thúc một nhiệm vụ mà drone không quay trở lại vị trí ban đầu. Khi người dùng xác nhận lấy vị trí, client sẽ gửi một request lên server, server sẽ thực hiện lệnh shell script gọi tới chương trình get_location của MAVSDK Program, get_location sẽ giao tiếp với drone và lấy dữ liệu về vị trí của drone, sau đó ghi vào file location trong Data Stores. Khi dữ liệu được đổ vào file location, Java Program sẽ lấy dữ liệu và trả về cho client, sau đó Bing Map API sẽ được sử dụng để tạo ra điểm tham chiếu dựa tên tọa độ trả về của drone và hiển thị ra giao diện người dùng.

Theo dõi drone: được sử dụng sau khi drone thực hiện hành động cất cánh và hạ cánh hoặc sau khi drone bắt đầu thực hiện nhiệm vụ. Khi người dùng xác nhận theo dõi, client sẽ gửi dữ liệu liên tục lên server mỗi 2 giây, bên phía server sẽ thực hiện lấy dữ liệu liên tục từ file output (nơi mà MAVSDK Program sẽ đổ dữ liệu vào khi nhận được dữ liệu từ drone), sau đó gửi về cho client. Bing Map API bên phía client sẽ tạo ra các điểm tham chiếu dựa vào vịt rí của drone được gửi về.

Ra lệnh cho drone thực hiện hành động cất cánh và hạ cánh: khi người dùng xác nhận thực hiện hành động cất cánh và hạ cánh, client sẽ gửi một request lên server, server sẽ thực hiện lệnh shell script gọi tới chương trình takeoff_and_land của MAVSDK Program, takeoff_and_land sẽ ra lệnh cho drone và lấy dữ liệu về vị trí của drone sau mỗi 2.5 giây (như đã trình bày ở MAVSDK Program), sau đó ghi vào file output trong Data Stores.

Ra lệnh cho drone thực hiện nhiệm vụ theo yêu cầu: ở phía client, người dùng sẽ thực hiện tạo nhiệm vụ cho drone bằng cách click vào các điểm trên bản đồ, người dùng cũng có thể nhập tọa độ bằng giá trị kinh độ và vĩ độ trực tiếp. Sau đó dữ liệu về nhiệm vụ sẽ được gửi lên server, server sẽ ghi dữ liệu này vào trong file input của Data Stores. Sau đó, nếu người dùng xác nhận thực hiện nhiệm vụ, client sẽ gửi một request lên server, server sẽ thực hiện lệnh shell script gọi tới chương trình fly_mission hoặc fly_mission_norl (do người dùng lựa chọn) của MAVSDK Program, fly_mission sẽ giao tiếp với drone và lấy dữ liệu về vị trí của drone, sau đó ghi vào file output trong Data Stores.

Sau quá trình xây dựng và thiết kế, em đã xây dựng được chương trình Web với giao diện như sau:

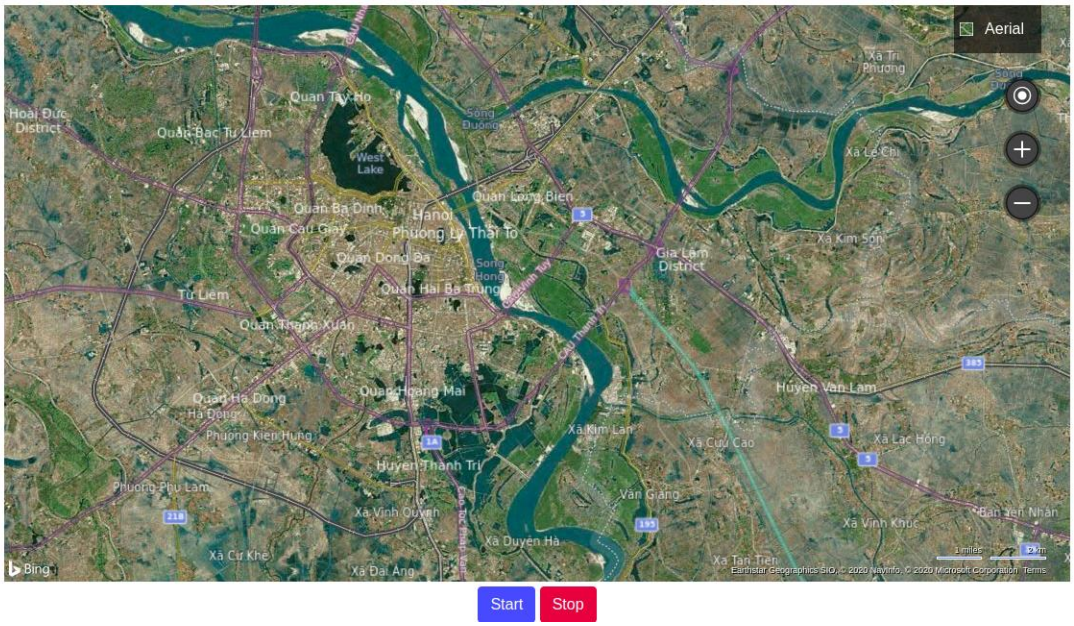


Hình 3.8 Giao diện các nút điều khiển và nhập liệu

Các nút với các chức năng như sau:

- **Connect**: tạo kết nối với server.
- **Disconnect**: hủy kết nối tới server.
- **Clear Data**: xóa dữ liệu ở client và server nhưng vẫn giữ danh sách nhiệm vụ.
- **Takeoff And Land**: ra lệnh cất cánh và hạ cánh cho drone.
- **Fly Mission RL**: thực hiện nhiệm vụ và quay về điểm ban đầu.

- **Fly Mission No RL**: thực hiện nhiệm vụ và hạ cánh tại điểm cuối cùng của nhiệm vụ.
- **Get Location**: lấy vị trí hiện tại của drone.
- **Start create mission**: cho phép người dùng tạo các điểm nhiệm vụ bằng cách click trên bản đồ.
- **Stop create mission**: đóng chức năng tạo điểm nhiệm vụ bằng click chuột (ta phải đóng chức năng này sau khi hoàn thành tạo các nhiệm vụ để tránh xảy ra lỗi không đáng có).
- **Upload mission**: tải nhiệm vụ lên server.
- **Clear mission**: Xóa nhiệm và và dữ liệu ở cả client và server.
- **Form nhập độ cao và tốc độ cho drone**: dùng để nhập tốc độ và độ cao cho drone, ấn Update để cập nhật dữ liệu lên server.
- **Form nhập tọa độ cho nhiệm vụ**: Cho phép người dùng nhập tọa độ cho nhiệm vụ, có thể kết hợp với tạo nhiệm vụ bằng click trên bản đồ.

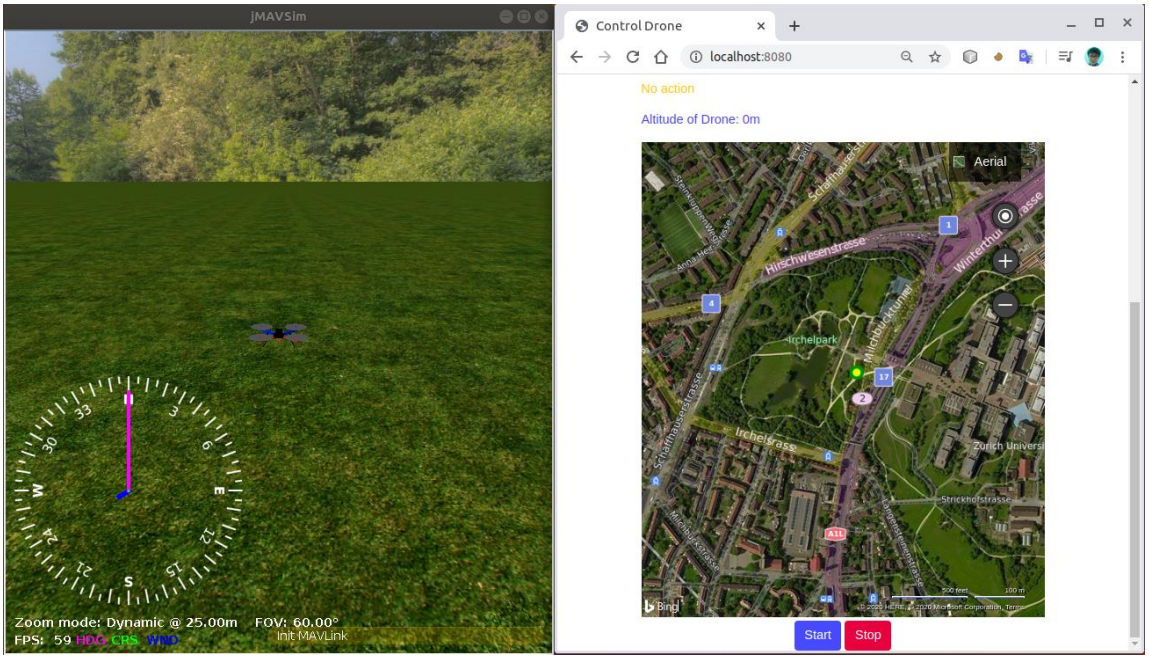


Hình 3.9 Giao diện để theo dõi và nhập nhiệm vụ cho drone

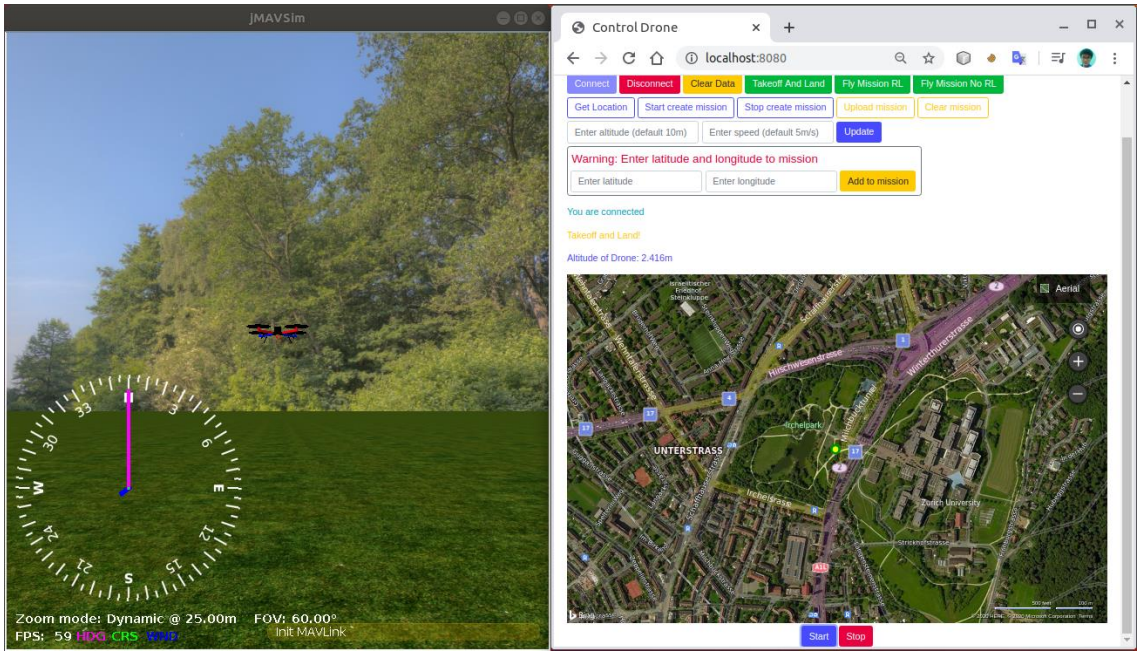
Trong đó:

- **Start**: bắt đầu theo dõi drone.
- **Stop**: dừng theo dõi drone.

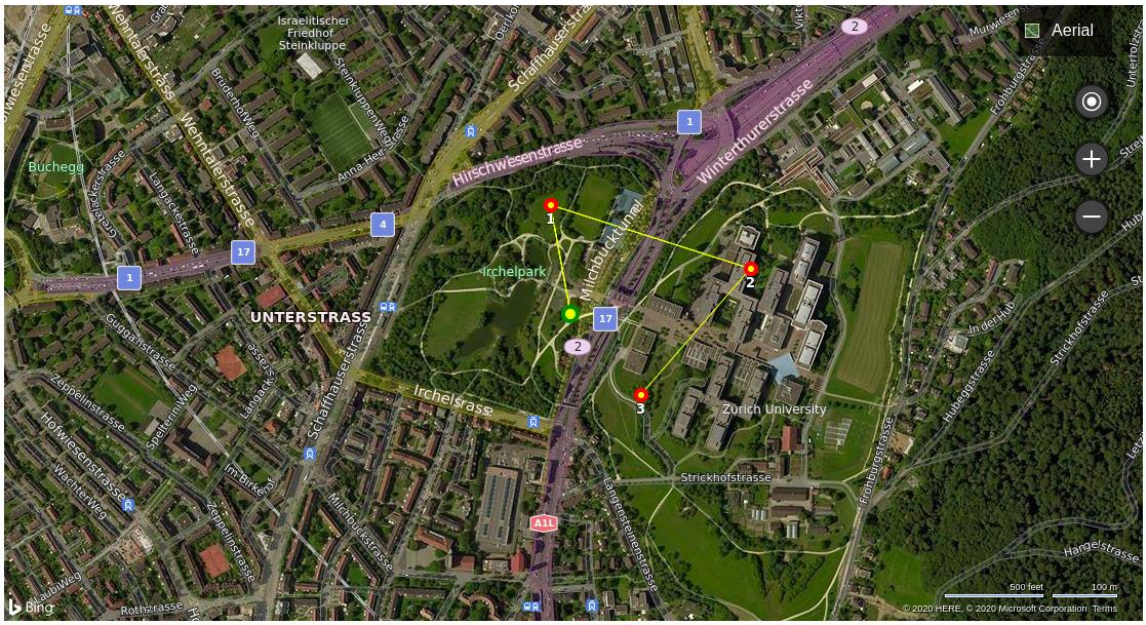
3.3 Kết quả chạy mô phỏng



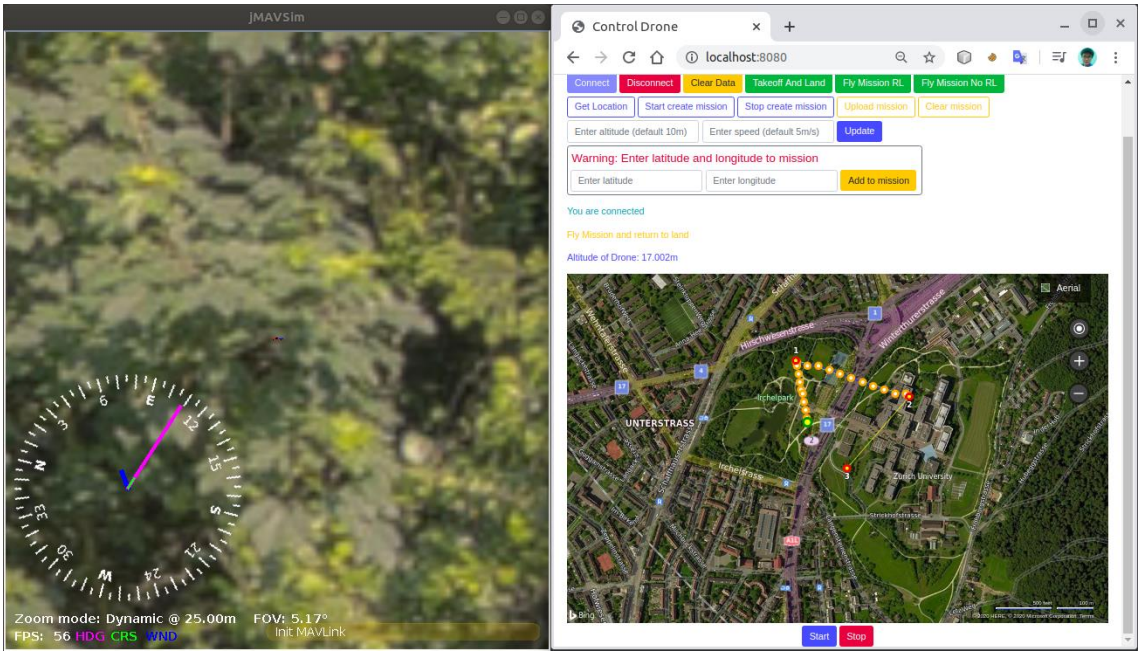
Hình 3.10 Giao diện map sau khi thực hiện Get location



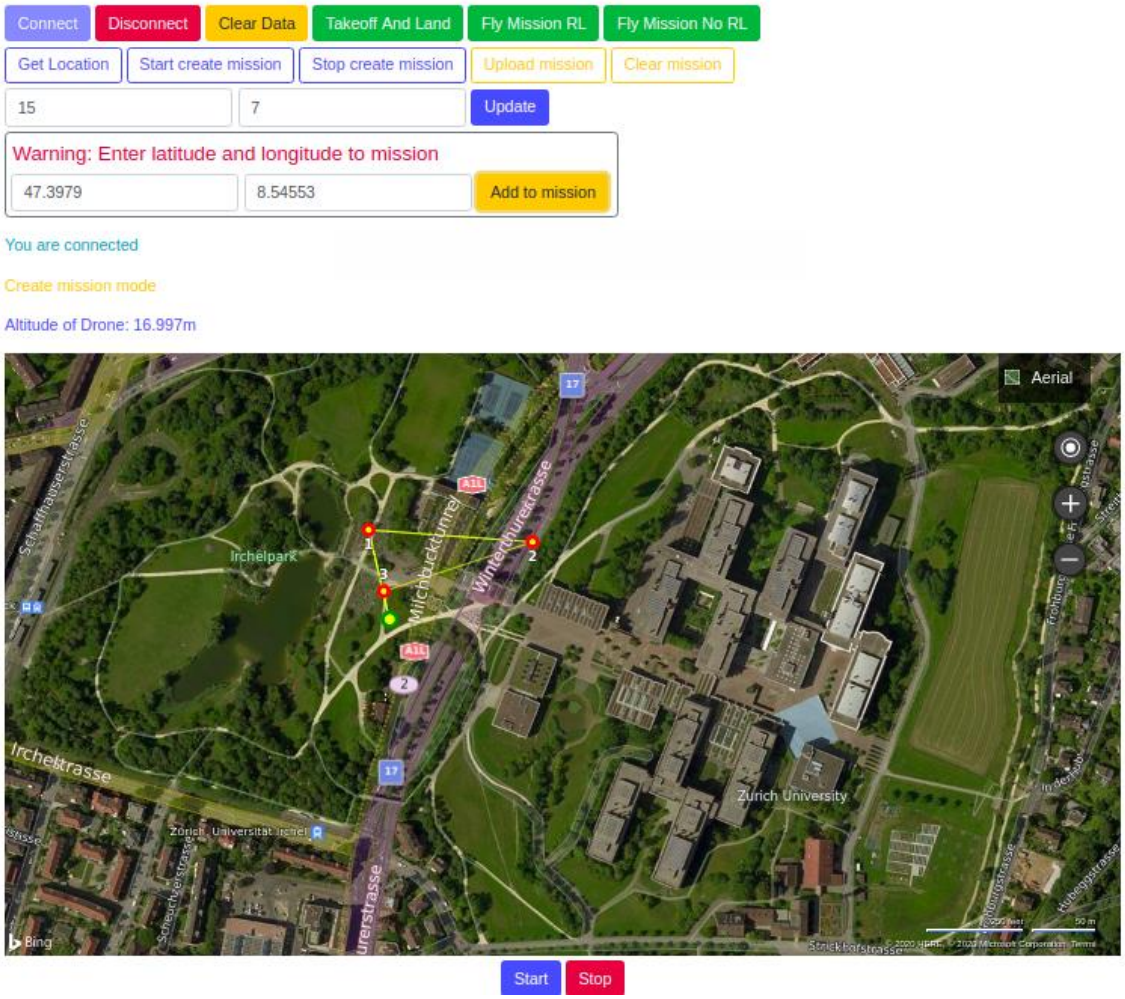
Hình 3.11 Giao diện sau khi chạy Takeoff And Land



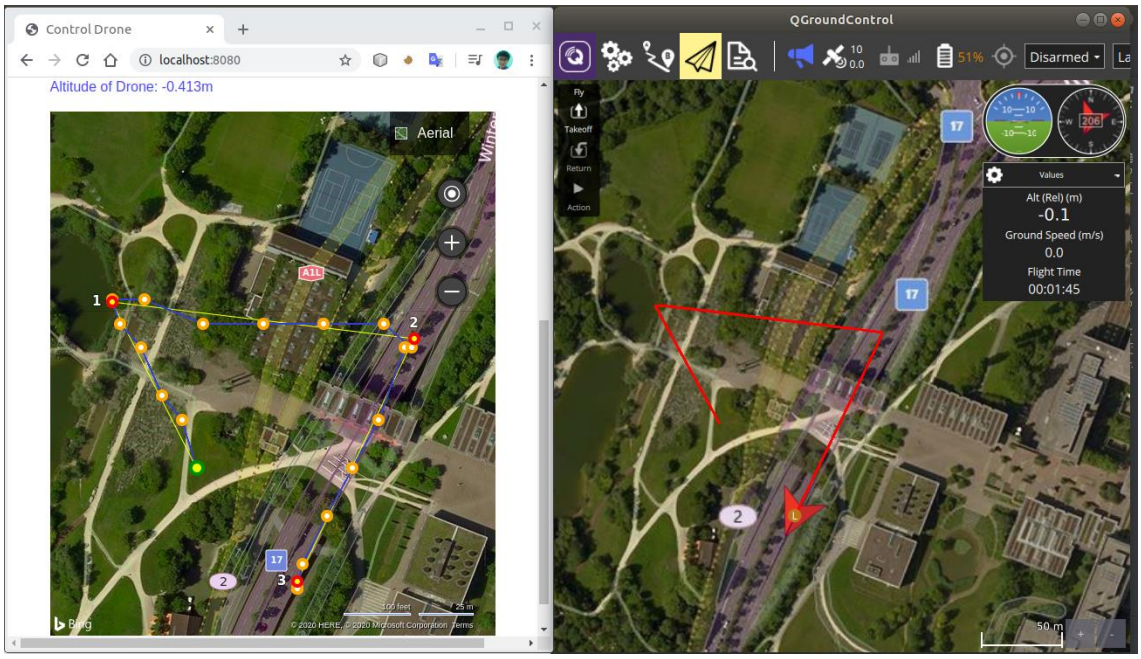
Hình 3.12 Giao diện map khi cài đặt các nhiệm vụ



Hình 3.13 Giao diện theo dõi khi drone thực hiện nhiệm vụ



Hình 3.14 Giao diện khi cài đặt nhiệm vụ bằng click trên bản đồ và nhập trực tiếp



Hình 3.15 So sánh với phần mềm QGroundControl

Như ta có thể thấy, đường bay thực tế của drone thường bị lệch so với đường bay thẳng giữa các điểm nhiệm vụ, điều này xảy ra có thể do rất nhiều yếu tố khách quan như môi trường, hay yếu tố chủ quan như định vị GPS không chính xác. Tuy nhiên ta thấy ở phần mềm QGroundControl, đường bay của drone lại rất chính xác, có thể do phần mềm QGroundControl có cơ chế tương thích đường bay thực tế với đường bay trên lý thuyết khi chênh lệch tọa độ không quá lớn. Tuy nhiên, trong nhiều hoạt động phức tạp và cần độ chính xác cao, thì việc thể hiện được chính xác tọa độ của drone sẽ là một yêu cầu quan trọng và không thể bỏ qua.

Như vậy, trong chương này, em đã trình bày cách xây dựng và thiết kế một phần mềm giao tiếp và quản lý thiết bị bay đơn giản có các chức năng: cập nhật vị trí của drone, ra lệnh cho drone thực hiện cất cánh và hạ cánh, cài đặt các nhiệm vụ cho drone và ra lệnh cho drone thực hiện nhiệm vụ theo yêu cầu, theo dõi drone khi drone cất cánh/hạ cánh và khi drone thực hiện nhiệm vụ.

KẾT LUẬN

Trên đây là bài báo cáo của em về đề tài : "Nghiên cứu trạm mặt đất ứng dụng giao tiếp và quản lý thiết bị bay". Trong bài báo cáo em đã tìm hiểu về cấu trúc và nguyên lý hoạt động cơ bản của drone, tìm hiểu về giao thức giao tiếp MAVLink giữa drone và trạm mặt đất cùng với MAVSDK – một thư viện với các API mạnh mẽ và dễ dàng sử dụng. Bên cạnh đó, em cũng đã nghiên cứu và xây dựng được một phần mềm (trạm mặt đất) dùng để giao tiếp và quản lý drone. Phần mềm đã đạt được một số yêu cầu nhất định: đã thực hiện được một số chức năng cơ bản như lấy về vị trí của drone, theo dõi drone và ra lệnh cho drone thực hiện cất cánh/hạ cánh, ra lệnh cho drone thực hiện nhiệm vụ theo yêu cầu, giao diện theo dõi xây dựng trên nền tảng Web dễ dàng sử dụng. Tuy nhiên, phần mềm còn một số hạn chế như: quy trình hoạt động còn phức tạp dẫn đến độ trễ cao, giao diện người dùng chưa xử lý được hết các lỗi nhập liệu dẫn đến yêu cầu người dùng phải thực hiện theo đúng yêu cầu đề ra. Hơn nữa, phần mềm hiện tại đang được chạy trên môi trường lý tưởng đó là sử dụng mô phỏng, trên thực tế vẫn còn rất nhiều yếu tố khác tác động đến qua trình điều khiển drone như: điều kiện thời tiết xấu, xuất hiện các vật cản, v.v. mà phần mềm hiện tại của em chưa xử lý được. Vì vậy, em kính mong thầy cô góp ý và cho em lời khuyên để em có thể cải thiện phần mềm của mình được hoàn thiện hơn. Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1] https://docs.qgroundcontrol.com/en/, truy cập cuối cùng ngày 16/6/2020.

[2] https://arxiv.org/pdf/1906.10641.pdf , truy cập cuối cùng ngày 16/6/2020.

[3] https://mavsdk.mavlink.io/develop/en/, truy cập cuối cùng ngày 16/6/2020.

[4] https://mavsdk.mavlink.io/develop/en/contributing/build.html, truy cập cuối cùng ngày 16/6/2020.

PHỤ LỤC

Phụ lục 1. File CmakeLists.txt của chương trình takeoff_and_land

```
cmake_minimum_required(VERSION 2.8.12)

project(takeoff_and_land)

if(NOT MSVC)
    add_definitions("-std=c++11 -Wall -Wextra -Werror")
else()
    add_definitions("-std=c++11 -WX -W2")
endif()

find_package(MAVSDK REQUIRED)

add_executable(takeoff_and_land
    takeoff_and_land.cpp
)

target_link_libraries(takeoff_and_land
    MAVSDK::mavsdk_telemetry
    MAVSDK::mavsdk_action
    MAVSDK::mavsdk
)
```

Phụ lục 2. File CmakeLists.txt của chương trình fly_mission

```
cmake_minimum_required(VERSION 2.8.12)

project(fly_mission)

find_package(Threads REQUIRED)

if(NOT MSVC)
    add_definitions("-std=c++11 -Wall -Wextra -Werror")
else()
    add_definitions("-std=c++11 -WX -W2")
endif()

find_package(MAVSDK REQUIRED)

add_executable(fly_mission
    fly_mission.cpp
)

target_link_libraries(fly_mission
    MAVSDK::mavsdk_action
    MAVSDK::mavsdk_mission
    MAVSDK::mavsdk_telemetry
    MAVSDK::mavsdk
    ${CMAKE_THREAD_LIBS_INIT}
)
```

Phụ lục 3. Mã nguồn của chương trình takeoff_and_land

```
#include <chrono>
#include <cstdint>
#include <mavsdk/mavsdk.h>
#include <mavsdk/plugins/action/action.h>
```

```
#include <mavsdk/plugins/telemetry/telemetry.h>
#include <iostream>
#include <thread>
#include <fstream>
#include <iomanip>

using namespace mavsdk;
using namespace std::this_thread;
using namespace std::chrono;

#define ERROR_CONSOLE_TEXT "\033[31m" // Turn text on console red
#define TELEMETRY_CONSOLE_TEXT "\033[34m" // Turn text on console blue
#define NORMAL_CONSOLE_TEXT "\033[0m" // Restore normal console colour

void usage(std::string bin_name)
{
    std::cout << NORMAL_CONSOLE_TEXT << "Usage : " << bin_name << "
<connection_url>" << std::endl
              << "Connection URL format should be :" << std::endl
              << " For TCP : tcp://[server_host][:server_port]" << std::endl
              << " For UDP : udp://[bind_host][:bind_port]" << std::endl
              << " For Serial : serial:///path/to/serial/dev[:baudrate]" <<
std::endl
              << "For example, to connect to the simulator use URL:
udp://:14540" << std::endl;
}

void component_discovered(ComponentType component_type)
{
    std::cout << NORMAL_CONSOLE_TEXT << "Discovered a component with type "
              << unsigned(component_type) << std::endl;
}

int main(int argc, char** argv)
{
    Mavsdk dc;
    std::string connection_url;
    connection_url = "udp://:14540";
    ConnectionResult connection_result;

    bool discovered_system = false;
    if (argc == 2) {
        connection_url = argv[1];
        connection_result = dc.add_any_connection(connection_url);
    } else {
        usage(argv[0]);
        return 1;
    }

    if (connection_result != ConnectionResult::Success) {
        std::cout << ERROR_CONSOLE_TEXT << "Connection failed: " <<
connection_result
                  << NORMAL_CONSOLE_TEXT << std::endl;
        return 1;
    }

    // We don't need to specify the UUID if it's only one system anyway.
    // If there were multiple, we could specify it with:
    // dc.system(uint64_t uuid);
    System& system = dc.system();

    std::cout << "Waiting to discover system..." << std::endl;
    dc.register_on_discover([&discovered_system](uint64_t uuid) {
```

```
        std::cout << "Discovered system with UUID: " << uuid << std::endl;
        discovered_system = true;
    });

    // We usually receive heartbeats at 1Hz, therefore we should find a system
after around 2
    // seconds.
    sleep_for(seconds(2));

    if (!discovered_system) {
        std::cout << ERROR_CONSOLE_TEXT << "No system found, exiting." <<
NORMAL_CONSOLE_TEXT
                  << std::endl;
        return 1;
    }

    // Register a callback so we get told when components (camera, gimbal)
etc
    // are found.
    system.register_component_discovered_callback(component_discovered);

    auto telemetry = std::make_shared<Telemetry>(system);
    auto action = std::make_shared<Action>(system);

    // We want to listen to the altitude of the drone at 1 Hz.
    const      Telemetry::Result      set_rate_result      =      telemetry-
>set_rate_position(1.0);
    if (set_rate_result != Telemetry::Result::Success) {
        std::cout  <<  ERROR_CONSOLE_TEXT  <<  "Setting  rate  failed:"  <<
set_rate_result
                  << NORMAL_CONSOLE_TEXT << std::endl;
        return 1;
    }

    float alt = -1;
    double lat = -1;
    double lon = -1;

    while(lat < 0 || lon < 0){
        const Telemetry::Position pos = telemetry->position();
        if(pos.latitude_deg > 0){
            alt = pos.relative_altitude_m;
            lat = pos.latitude_deg;
            lon = pos.longitude_deg;
        }
    }

    std::cout << "Altitude: " << alt << " - Latitude: " << lat << " -
Longitude: " << lon
              << std::endl;

    while (telemetry->armed()) {
        // Wait until we're done.
        sleep_for(seconds(1));
    }
    std::cout << "Disarmed, exiting." << std::endl;

    std::ofstream myfile;

myfile.open("/home/hieu/MAVSDK_Project_Test/data/location",std::ios::app);
    myfile << alt << ",";
    myfile << lat << ",";
    myfile << lon << "\n";
```

```
    myfile.close();

    return 0;
}
```

Phụ lục 4. Một số công cụ được sử dụng trong phần mềm

Make: GNU Make là một công cụ kiểm soát việc tạo ra các file thực thi và những file khác (không là file mã nguồn) của một chương trình từ mã nguồn của chương trình đó.

Make có thể hiểu được cách để build phần mềm từ một file gọi là makefile, trong đó liệt kê những file đích và cách để làm ra những file đó từ các file khác.

Make giúp người dùng cuối có thể build và cài đặt gói phần mềm mà không cần để ý tới các bước cụ thể – bởi những chi tiết đó đã nằm trong makefile.

Make tìm ra những file nào sẽ được cập nhật, dựa trên file nguồn nào đã được thay đổi. Và cũng xác định trình tự cập nhật các file trong trường hợp file đích này phụ thuộc vào một file đích kia. Vậy nên nếu chúng ta thay đổi vài dòng trong mã nguồn và sau đó chạy Make, ta sẽ không cần biên dịch lại toàn bộ chương trình. Make chỉ thay đổi những file đích liên quan đến file nguồn mà ta thay đổi.

Make không giới hạn về bất cứ ngôn ngũ lập trình nào. Trong makefile sẽ chỉ đến shell (shell là giao diện tới service của hệ điều hành) để xử lý công việc. Những câu lệnh này có thể gọi một trình dịch để tạo ra các file object, gọi linker để tạo ra một file thực thi (link các object), hoặc gọi ar để cập nhật một thư viện, hoặc gọi TeX hay Makeinfo để cập nhật tài liệu phần mềm.

Makefile và Cmake: makefile như là một file chứa các công thức mà Make sẽ chạy. Đối với makefile, ta có thể hiểu đơn giản rằng những thứ ở trong makefile bao gồm các Target và Dependency. Target là một đối tượng cần được "make", Dependency là những thứ cần thiết để tạo nên Target đó. Phần mềm của chúng ta sẽ có hàng loạt những Target và Dependency chồng chéo hoặc không chồng chéo nhau để tạo ra sản phẩm đích, ta chỉ cần liệt kê ra, Make sẽ biết đúng trình tự để thực hiện công việc. Cấu trúc của một rule đơn giản trong makefile gồm có dependencies và commands. Ở đây commands chính là những câu lệnh để tiến hành xây dựng target. Khi gọi Make, ta có thể chỉ ra target cụ thể để Make chạy, hoặc Make sẽ tự chọn target được liệt kê đầu tiên (dĩ nhiên nếu target đó phụ thuộc (depends on) vào một target khác thì những target đó phải được cập nhật trước).

Tiện ích make và Makefiles cung cấp một hệ thống build mà chúng ta có thể sử dụng để quản lý việc compile và re-compilation của một chương trình được viết bằng ngôn ngữ bất kỳ. Việc sử dụng Makefiles đôi khi lại có thể trở thành một công việc phức tạp trong trường hợp project mà chúng ta build có nhiều sub directories hoặc sẽ phải triển khai trên nhiều nền tảng khác nhau.

Để khắc phục điều đó thì CMake ra đời. CMake là một công cụ sinh Makefile đa nền tảng. Nói đơn giản thì CMake sẽ tự động tạo ra Makefiles cho project của chúng ta. Trong đồ án này, em sử dụng Cmake để xây dựng các chương trình được viết bằng C++ dùng để giao tiếp và quản lý drone.

WebSocket: Websocket là giao thức hỗ trợ giao tiếp hai chiều giữa client và server để tạo một kết nối trao đổi dữ liệu. Giao thức này không sử dụng HTTP mà thực hiện nó qua TCP. Mặc dù được thiết kế để chuyên sử dụng cho các ứng dụng web, lập trình viên vẫn có thể đưa chúng vào bất kì loại ứng dụng nào.

Ưu điểm: WebSocket cung cấp giao thức giao tiếp hai chiều mạnh mẽ. Nó có độ trễ thấp và dễ xử lý lỗi. Websocket thường được sử dụng cho những trường hợp yêu cầu real time như chat, hiển thị biểu đồ hay thông tin chứng khoán.

Các gói tin (packets) của Websocket nhẹ hơn HTTP rất nhiều. Nó giúp giảm độ trễ của network nhiều lần.

STOMP: STOMP (Streaming Text Oriented Messaging Protocol): (Giao thức luồng văn bản theo hướng tin nhắn) là một giao thức truyền thông, một nhánh của WebSocket. Khi client và server liên lạc với nhau theo giao thức này chúng sẽ chỉ gửi cho nhau các dữ liệu dạng tin nhắn văn bản. Mối quan hệ giữa STOMP và WebSocket cũng gần giống mối quan hệ giữ HTTP và TCP.

Vậy, tại sao sử dụng STOMP nếu chúng ta đã sử dụng WebSocket ? Hoặc ngược lại, tại sao sử dụng WebSocket nếu chúng ta đang sử dụng STOMP ?

STOMP mô tả định dạng tin nhắn được trao đổi giữa máy khách và máy chủ. Mặt khác, WebSocket không có gì ngoài một giao thức giao tiếp.

Chúng ta không thể "chỉ sử dụng" STOMP để liên lạc với máy chủ message broker. Chúng ta phải sử dụng một phương tiện để gửi các tin nhắn STOMP đó, một trong số đó là WebSocket.

STOMP không quan tâm đến cái bắt tay WebSocket (WebSocket handshake), trên thực tế, nó hoàn toàn không biết về nó. Chúng ta có thể sử dụng STOMP trên một giao thức truyền tải khác (ví dụ: HTTP) và không thấy sự khác biệt nào từ phối cảnh STOMP.

Tính năng chỉ muốn gửi tin nhắn cho người dùng đã đăng ký một chủ đề cụ thể hoặc gửi tin nhắn đến một người dùng cụ thể khó thực hiện hơn với WebSocket đơn giản , nhưng STOMP có tất cả các tính năng này, vì nó được thiết kế để tương tác với message broker.

Trong phạm vi đồ án này, em sử dụng STOMP để gửi và nhận dữ liệu về vị trí (độ cao, kinh độ, vĩ độ) tới Data stores.