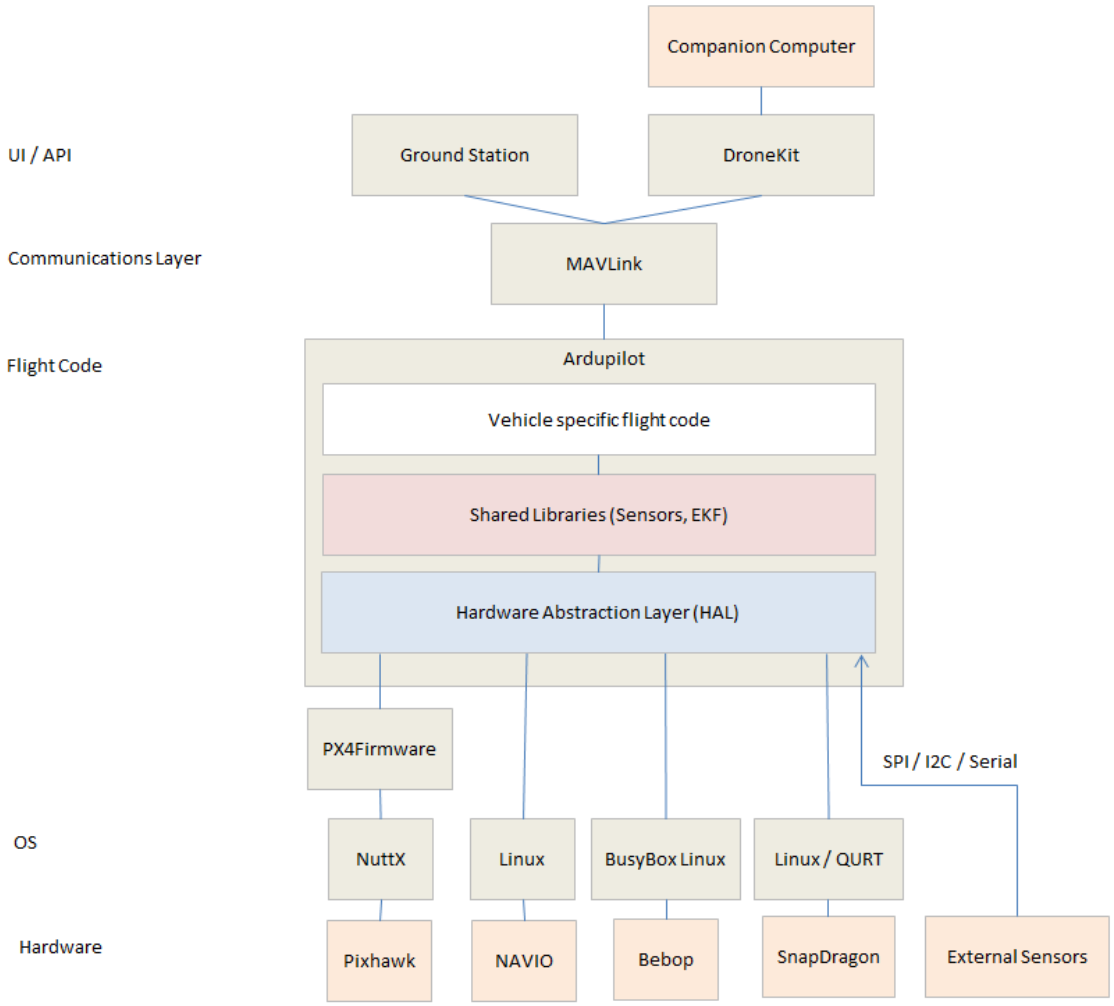
Giới thiệu về ardupilot và arducopter

Ardupilot là một open source ban đầu được tạo ra để điều khiển quadcopter, repository của Ardupilot được tạo vào tháng 11/2009 bởi Jordi Munoz. Hiện nay, Ardupilot bao gồm nhiều module sử dụng được cho nhiều loại máy bay khác nhau với các bo điều khiển khác nhau. ArduCopter là một module trong Ardupilot, dùng để điều khiển các máy bay lên thẳng như Quadcopter, Hexcopter, SingleCopter, Helicopters, … hiện nay Arducopter đã có tới version 3.5.

Kiến trúc tổng quát của ardupilot và quan hệ với các thành phần khác

Ardupilot chứa source code để điều khiển máy bay, xe mô hình, anten; có thể clone hoặc download về máy tính từ dùng git từ địa chỉ https://github.com/ArduPilot/ardupilot.git



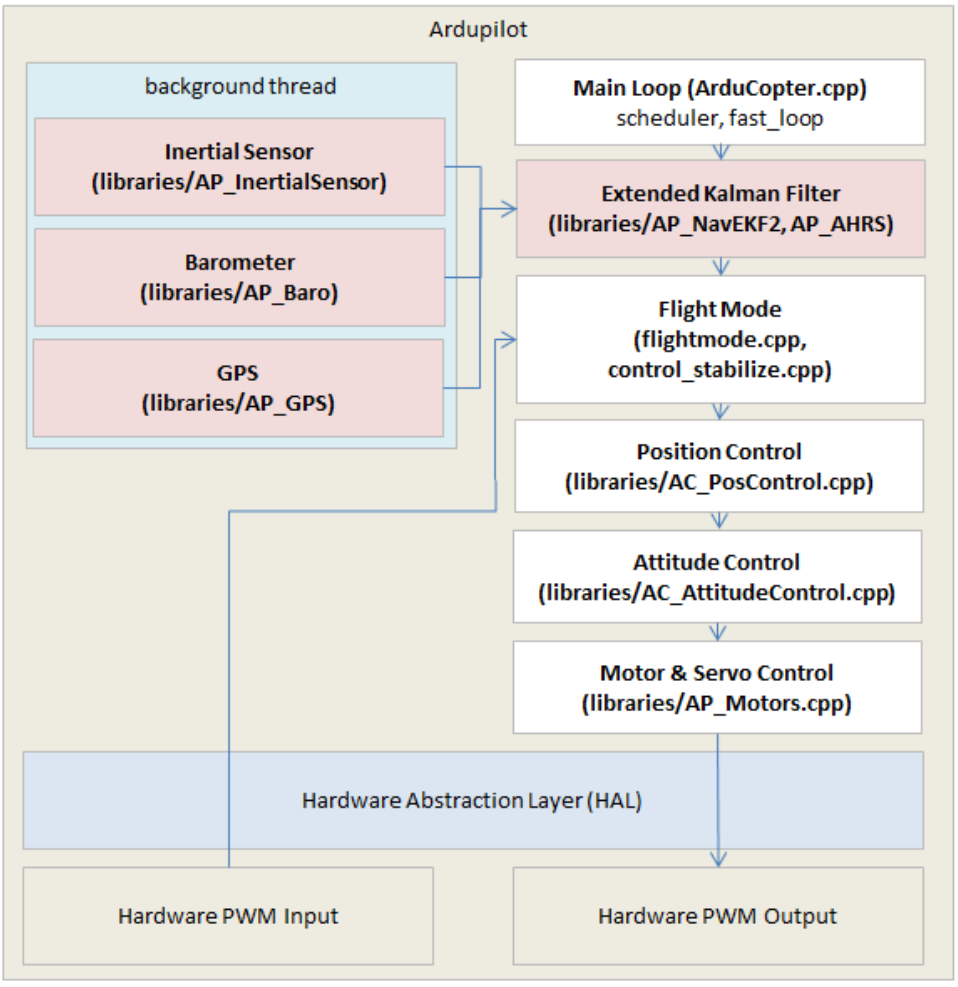
Hình 1 Cấu trúc tổng quan soure code Ardupilot

Source code ardupilot gồm các năm phần chính:

- Code điều khiển cho các loại máy bay khác nhau, xe mô hình, điều khiển anten (folder Arducopter: máy bay lên thẳng, ArduPlane: máy bay cánh bằng, APMRover: xe mô hình, AntennaTracker: điều khiển anten).
- Các thư viện chung: Các thư viện giao tiếp ngoại vi, đọc, lọc nhiễu cảm biến (các folder AP_... nằm trong foler libraries). Thư viện thực hiện các bộ điều khiển (các folder AC_... nằm folder libraries). Chức năng cụ thể của từng thư viện trong folder libraries tham khảo trong link sau http://ardupilot.org/dev/docs/apmcopter-programming-libraries.html#apmcopter-programming-libraries

- AP_HAL: Ardupilot có thể chạy trên nhiều platform khác nhau, với mỗi platform Ardupilot cung cấp các file để giao tiếp với phần cứng và sử dụng scheduler của hệ điều hành tương ứng. Folder AP_HAL define class cha cho các class con ứng với từng platform: AP_HAL_AVR cho bo AVR, AP_HAL_PX4 cho Pixhawk và AP_HAL_Linux cho bo nhúng Linux.
- Tools: chứa các công cụ hỗ trợ như tools/autotest chứa phần mềm test các module.
- Các source code hỗ trợ khác: PX4NuttX, PX4Firmware, uavcan, mavlink: các file thư viện mavlink chứa các method để truyền và phân tích gói Mavlink từ MissionPlanner, Dronekit hay phần mềm khác.

Thứ tự thực hiện chương trình trong Ardupilot



Hình 2 Thứ tự thực hiện chương trình trong Ardupilot

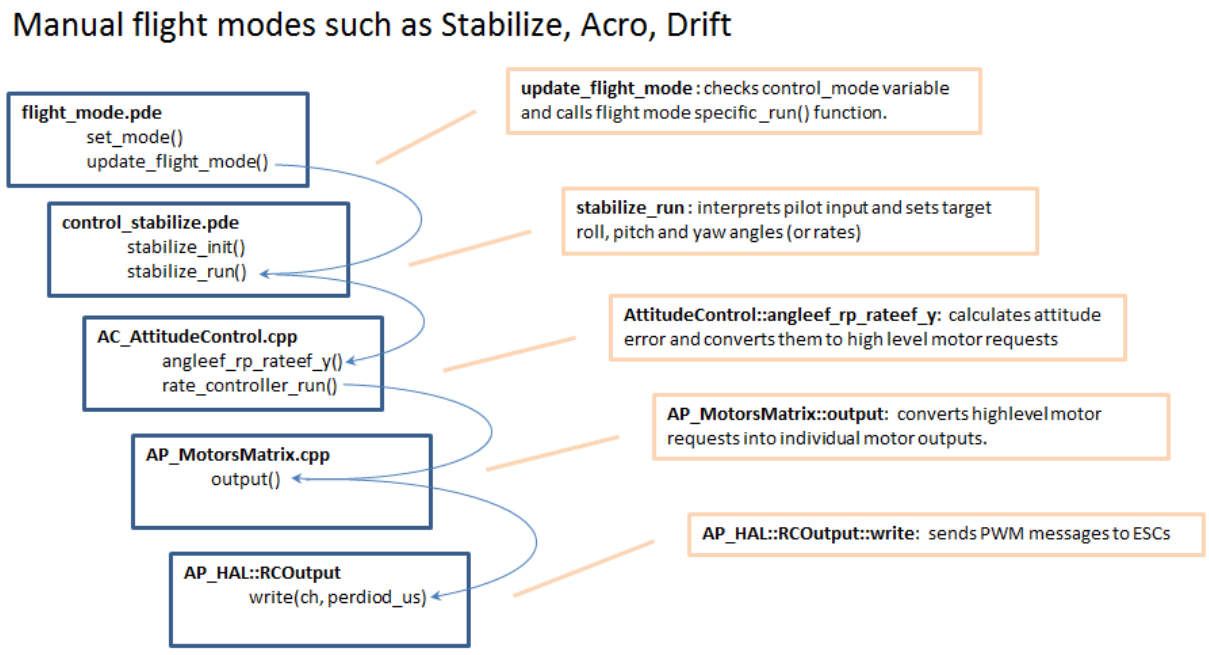
Trong file ArduCopter/ArduCopter.cpp có method *loop()*, method này được gọi với tần số 400 Hz, có chức năng như loop while(1) trong hàm *main()* thông thường.

Method *fast_loop()* được gọi trong method *loop()*, method này có chức năng thực hiện bộ điều khiển vận tốc góc (*rate_controller_run()*) và xuất giá trị ra các ESC (*motors_output()*), cập nhật giá trị cảm biến và thực hiện bộ lọc Kalman (*ins.update(), read_AHRS()*), update các mode bay (*update_flight_mode()*).

Method *update_flight_mode()* sẽ gọi method *xxx_run()* trong file ArduCopter/control_xxx.cpp với xxx là mode bay (stabilize, AltHold, …).

Tùy vào mode bay, ta sẽ cài đặt vị trí x,y,z mong muốn bằng các method trong file libraries/AC_PosControl.cpp hoặc giá trị góc đặt mong muốn trong file libraries/AC_Attitude.cpp, giá trị góc đặt này có thể lấy từ tay cầm (*channel_roll->get_control_in(), channel_pitch->get_control_in(), channel_yaw->get_control_in(), channel_throttle->get_control_in()*) hoặc từ bộ điều khiển vị trí. Chi tiết về code điều khiển được trình bày trong file "Ardupilot_Controller".

❖ Ví dụ: mode bay Stabilize



Hình 3 Thứ tự thực hiện method update_flight_mode với mode bay stabilize.

Scheduler

Trong file ArduCopter.cpp ta thấy có mảng *scheduler_tasks[]*, mỗi phần tử của mảng này là một tác vụ sẽ được chạy với tần số nhất định.

- Ví dụ: task SCHED_TASK(rc_loop, 100, 130) nghĩa là task rc_loop sẽ được chạy với tần số 100 Hz và thời gian tối đa task này chạy là 130 us.

Nếu ta muốn chạy một tác vụ định kì thì có thể sử dụng các task *userhook* trong mảng *scheduler_tasks[]*: Ardupilot cung cấp các method userhook *userhook_FastLoop, userhook_50Hz, userhook_MediumLoop, userhook_SlowLoop, userhook_SuperSlowLoop* lần lượt chạy ở tần số 100 Hz, 50 Hz, 10 Hz, 3.3 Hz và 1 Hz.

Để enable các method userhook ta uncomment các define trong ArduCopter/APM_Config.h

```
//#define USERHOOK_INIT userhook_init();
//#define USERHOOK_FASTLOOP userhook_FastLoop();
//#define USERHOOK_50HZLOOP userhook_50Hz();
//#define USERHOOK_MEDIUMLOOP userhook_MediumLoop();
//#define USERHOOK_SLOWLOOP userhook_SlowLoop();
//#define USERHOOK_SUPERSLOWLOOP userhook_SuperSlowLoop();
```

và code các method này trong file ArduCopter/UserCode.cpp

Class Copter

Được định nghĩa trong file Copter.h. Các method của class này được hiện thực trong các file của folder ArduCopter, bắt đầu bởi *Copter::*.

- Ví dụ trong file control_althold.cpp có method *bool Copter::althold_init(bool ignore_checks).*

Ý nghĩa và cách điều khiển các mode bay thông dụng:

Stabilize: 3 kênh roll, pitch, yaw của tay cầm điều khiển góc nghiêng roll, pitch, yaw của drone và kênh ga điều khiển tốc độ trung bình của 4 động cơ.

AltHold: tay cầm điều khiển góc nghiêng roll, pitch, yaw của động cơ; kênh ga điều khiển tốc độ đi lên/xuống của drone.

Loiter: drone giữ vị trí 3D, góc nghiêng, mode bay này cần có cảm biến GPS. Kênh roll, pitch điều khiển gia tốc tịnh tiến trục x, y; kênh ga điều khiển tốc độ đi lên/xuống như mode AltHold; kênh yaw điều khiển tốc độ quay yaw.

Autotune: tune thông số PID cho bộ điều khiển roll/pitch/yaw.

Tài liệu tham khảo:

[1] http://ardupilot.org/dev/docs/learning-ardupilot-introduction.html