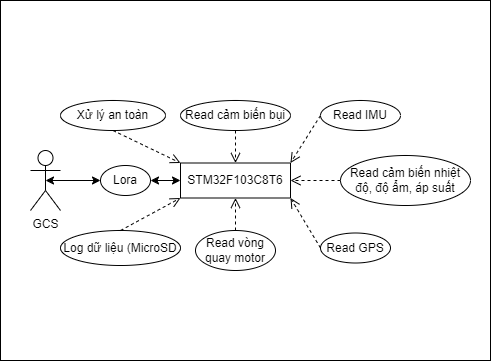
SYSTEM ARCHITECTURE DOCUMENT

# Giới thiệu và mục tiêu

Tài liệu này mô tả chức năng và nhiệm vụ cho vi điều khiển STM32F103C8T6 được lắp đặt trên vệ tinh cansat.

Phạm vi nghiên cứu tập trung vào hệ thống xử lý thời gian thực cho các cảm biến được lắp đặt trên thân của Cansat, có cơ chết giao tiếp với trạm mặt đất qua lora và lưu dữ liệu vào module thẻ nhớ trực tiếp trên Cansat.

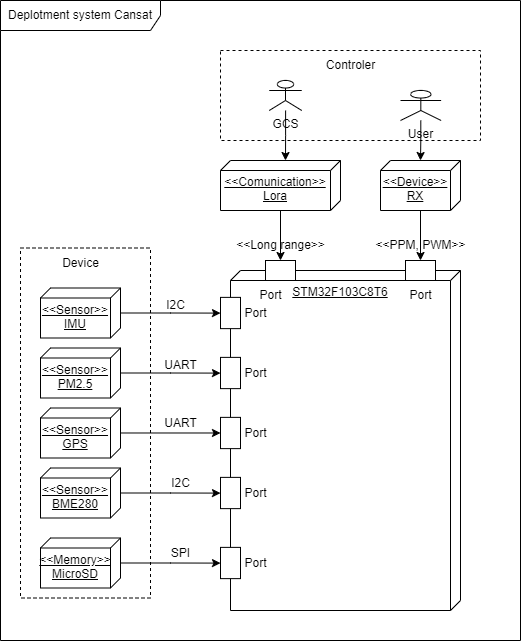
## Yêu cầu



Mô tả hoạt động của hệ thống

|  |  |  |
| --- | --- | --- |
| ID | Yêu cầu | Giải thích |
| R1: | Đọc và log dữ liệu | Đọc và xử lý các giá trị cảm biến trên Cansat và log dữ liệu vào module thẻ nhớ |
| R2: | Giao tiếp trạm mặt đất | Giao tiếp với trạm mặt đất qua Lo-ra, dữ lieu được gửi về là các giá trị cảm biến đã qua xử lý |
| R3: | Hệ thống an toàn | Có cơ chế an toàn khi động Cansat hoạt động không đúng kế hoạch và hoạt động không an toàn. |

# Bối cảnh kỹ thuật



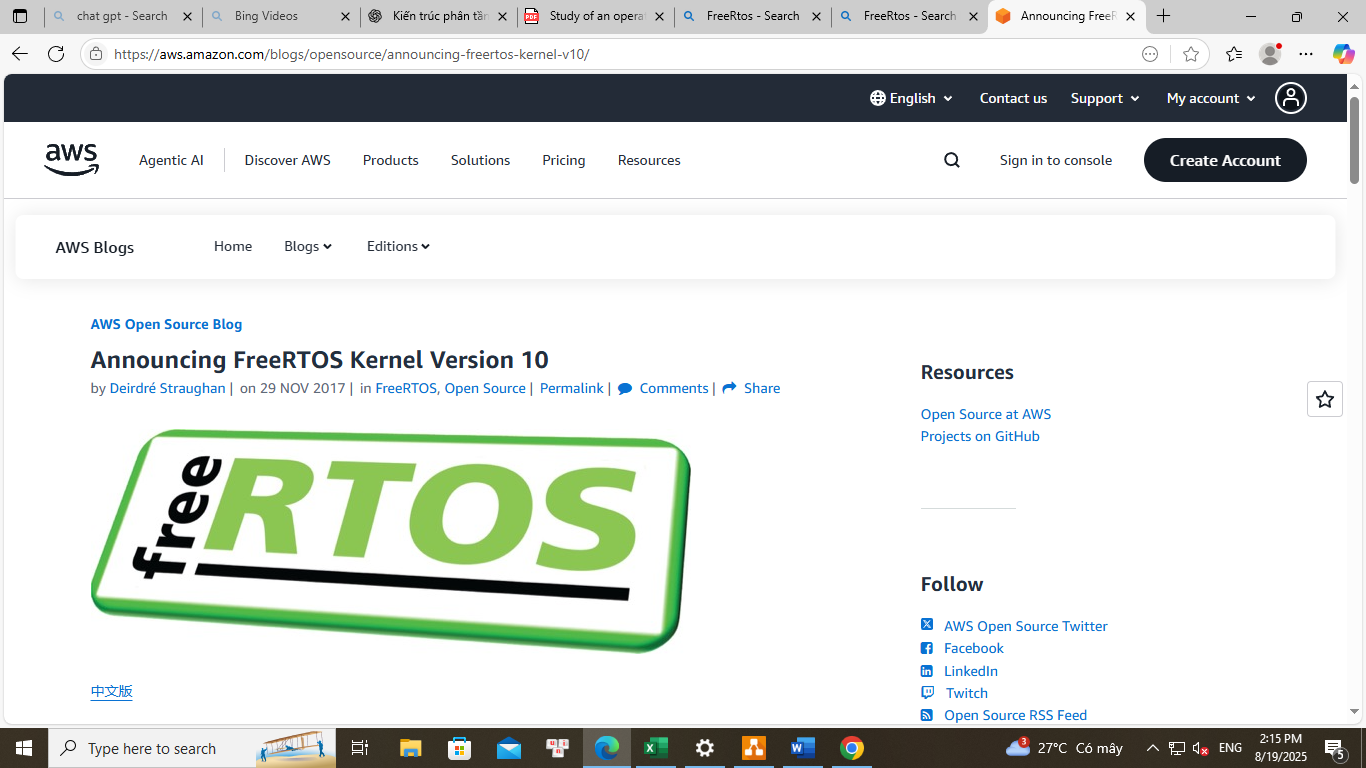
Các thành phần tương tác với hệ thống

|  |  |
| --- | --- |
| Phần ngoài control system | Mô tả |
| Controler | Gửi lệnh điều khiển đến STM32F103 cho mục đích gửi chế độ phóng dù khi cần thiết, giao tiếp thông số trạng thái |
| Device | Bao gồm các sensor và module lưu trữ dữ liệu |

# Kiến trúc phần mềm

## Giới thiệu về FreeRTOS

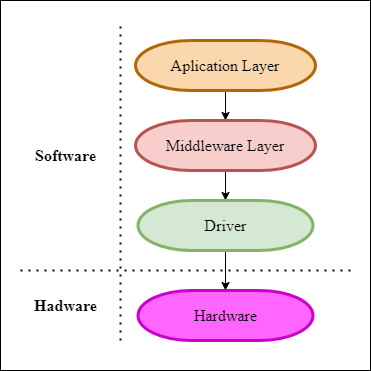
FreeRTOS là một hệ điều hành thời gian thực (Real-Time Operating System – RTOS) được ứng dụng nhiều trong hệ thống nhúng. FreeRTOS cho phép vi điều khiển chay nhiều tác vụ (task) song song với nhau một các độc lập, các task có thể chạy định kì thông qua Timer software gọi hàm callback hoặc kích hoạt các task theo chu kỳ [1]. Hệ điều hành cũng hỗ trợ quản lý task, queue, semaphore, mutex, timer và các cơ chế đồng bộ khác, gúp trao đổi dữ liệu giữa các task một cách an toàn và hiệu quả[1].



Logo hệ điều hành FreeRTOS

Ứng dụng thực tế, FreeRTOS được sử dụng để tổ chức các vụ song song trong hệ thống, giúp tăng tính ổn định và đáp ứng thời gian thực cho các chức năng trong khối cảm biến và khối điều khiển chính. Do cấu trúc firmware được thiết kế theo kiểu module do đó dễ dàng them bớt các task nhiệm vụ mà không ảnh hưởng đến hệ thống tổng thể, giúp mã nguồn dễ dàng tái sử dụng và mở rộng cho nhiều ứng dụng khác nhau.

## Kiến trúc chung của cà 2 khối phần mềm



Kiến trúc phân tầng hệ thống

Kiến trúc chung của hệ thống phần mềm được chia thành bốn tầng: Application, Middleware, Driver và Hardware.

|  |  |
| --- | --- |
| Tầng | Chức năng |
| Tầng Hardware | Bao gồm các phần cứng trực tiếp giao tiếp với hệ thống |
| Tầng Driver | Tách biệt phần mềm khỏi phần cứng cụ thể bằng cách cung cấp các API chuẩn, cho phép truy cập và điều khiển phần cứng một cách thống nhất. |
| Tầng Middleware | Cung cấp các dịch vụ chung, làm cầu nối giữa tầng Application và tầng Driver. Trong FreeRTOS, tầng Middleware chịu trách nhiệm quản lý các task, lập lịch, giao tiếp giữa các task, đồng thời cung cấp API giúp tầng Application triển khai chức năng dễ dàng và mở rộng. |
| Tầng Application | Xử lý các chức năng chính của hệ thống. Các chức năng này được triển khai dưới dạng các task cụ thể. |

Danh sách chức năng của các tầng hệ thống

Trong dự án này, trọng tâm nghiên cứu và phát triển là hai tầng Middleware và Application, nơi đảm nhận việc quản lý tài nguyên, điều phối task và xử lý logic chính của hệ thống [2].

## Driver

Mục này giới thiệu các giao thức được sử dụng trong dự án.

### UART

### I2C

### PWM

### PPM

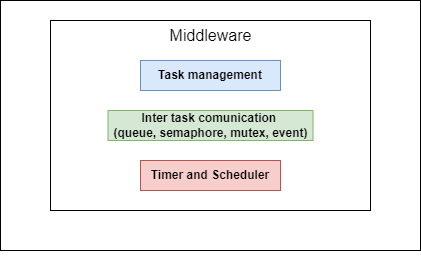
### Mavlink

## Middleware

### FreeRTOS trong middleware

Middleware là tầng trung gian giữa tầng application và tầng Driver, cho phép tầng application tách biệt với phần drive, cung cấp các phương thức giao tiếp làm đơn giản hóa cho tầng application.

FreeRTOS được đơn giản hóa trong lớp middleware, có nhiệm vụ tạo và quản lý task, giao tiếp giữa các task, định thời, trừu tượng hóa phần cứng.



Mô tả chức năng trong Middleware

Task management cung cấp giao diện tạo và quản lý task, bao gồm: đặt độ ưu tiên, stack size từng task, quán lý danh sách stack và trạng thái.

Inter task communication: khởi tạo và cung cấp các cơ chế giao tiếp an toàn giữa các task queue; tạo API cho phép đọc dữ liệu đơn giản hóa trong tầng application.

Timer và Scheduler định thời cho các task theo chu kỳ cố định, cung cấp chế độ một lần hoặc chu kỳ để callback hoặc kích hoạt task.

## Chi tiết lớp Application của khối cảm biến

|  |  |  |
| --- | --- | --- |
| Tên Task | Chức năng | Tần số |
| IMUTask() | Điều chế xung cho motor của Rover | 50 HZ |
| LoraTask() | Đóng gói và giao tiếp tới trạm mặt đất | 1 HZ |
| PM25Task() | Đo cảm biến bụi trong không khí | 1 HZ |
| BME280Task() | Đo nhiệt độ, áp suất, đọ ẩm từ môi trường | 5 HZ |
| MicroSDTask() | Log dữ liệu trong quá trình bay | 1 HZ |
| GPSTask() | Lấy dữ liệu từ cảm biến GPS | 5 HZ |
| PIDTask() | Cân bằng | 50 |
| MotorTask() | Điều khiển động cơ | 50 |

Các task thực thi