# BÀI TẬP MINI PROJECT: QUẢN LÝ TRẠNG THÁI MODULE TRONG HỆ THỐNG Ô TÔ

**Mục tiêu:**
Học viên sẽ thiết kế và triển khai một hệ thống quản lý trạng thái các module trong ô tô, sử dụng các kỹ thuật lập trình C quan trọng như bitmask, struct & union, function pointer, setjmp/longjmp, và cấp phát động (malloc/free).

Bài tập này giúp sinh viên hiểu cách quản lý và điều khiển các module như động cơ (ECU), phanh (ABS), đèn, điều hòa, cảm biến nhiệt độ trong xe hơi.

## I. MÔ TẢ CHI TIẾT PROJECT

### 1. Bối cảnh bài toán

Hệ thống ô tô hiện đại có nhiều ECU (Electronic Control Unit) để điều khiển các chức năng khác nhau. Mỗi ECU đại diện cho một module (động cơ, hệ thống phanh, cảm biến,...) và các module này có trạng thái ON/OFF, lỗi, cảnh báo.

Trong project này, học viên sẽ xây dựng một chương trình mô phỏng việc quản lý và kiểm soát trạng thái các module trong ô tô. Chương trình phải có khả năng:

- Thêm/Xóa module vào danh sách (dùng cấp phát động).
- Quản lý trạng thái ON/OFF, lỗi, cảnh báo (bitmask).
- Sử dụng function pointer để điều khiển module.
- Dùng setjmp/longjmp để xử lý lỗi khi module gặp sự cố.
- Tối ưu lưu trữ dữ liệu module bằng struct và union.

### 2. Các thành phần chính

**Struct quản lý module**

- Mỗi module có ID, tên, trạng thái, quyền truy cập, hành động điều khiển.
- Trạng thái module dùng bitmask để lưu ON/OFF, lỗi, cảnh báo.
- Mảng động chứa danh sách các module (dùng malloc/realloc/free).

**Bitmask để quản lý trạng thái**

- Trạng thái ON/OFF, lỗi, cảnh báo được lưu bằng bitmask:

```
#define STATUS_ON         (1 << 0)  // Module đang bật
#define STATUS_ERROR      (1 << 1)  // Module gặp lỗi
#define STATUS_WARNING    (1 << 2)  // Cảnh báo lỗi
```

**Function Pointer để điều khiển module**

- Dùng con trỏ hàm để thực hiện bật/tắt, kiểm tra trạng thái module.
- Mỗi module có thể có các hành động riêng.

**Xử lý lỗi bằng Setjmp/Longjmp**

- Nếu module gặp lỗi, hệ thống sẽ nhảy về trạng thái an toàn mà không dừng chương trình.

## II. CẤU TRÚC THƯ MỤC VÀ FILE

```
Automotive_Module_Manager/
├── main.c               // Chương trình chính
├── module_manager.c     // Xử lý danh sách module
├── module_manager.h     // Header file của module
manager
├── bitmask_utils.c      // Hàm hỗ trợ bitmask
├── bitmask_utils.h      // Header file cho bitmask
├── function_handler.c   // Xử lý con trỏ hàm
├── function_handler.h   // Header file của function
handler
├── error_handler.c      // Xử lý lỗi
├── error_handler.h      // Header file xử lý lỗi
```

## III. CÁC BƯỚC THỰC HIỆN

**Bước 1: Phân tích yêu cầu**

- Hiểu cách hệ thống module trên ô tô hoạt động.
- Xác định danh sách module sẽ quản lý: Động cơ, ABS, Đèn, Điều hòa, Cảm biến nhiệt độ...
- Xác định trạng thái của mỗi module (ON/OFF, lỗi, cảnh báo).

- Lập kế hoạch quản lý module bằng mảng động.

**Bước 2: Thiết kế cấu trúc dữ liệu**

- Thiết kế struct Module để lưu thông tin module.
- Dùng Union để tối ưu bộ nhớ trạng thái.

**Bước 3: Triển khai chức năng quản lý module**

- Cấp phát động để lưu danh sách module.
- Hàm thêm module.
- Hàm xóa module.
- Hàm kiểm tra trạng thái module.

**Bước 4: Xử lý lỗi với Setjmp/Longjmp**

- Nếu module gặp lỗi, chương trình quay về trạng thái an toàn.

**Bước 5: Vẽ Flowchart**

- Sinh viên tự vẽ flowchart theo quy trình.
- Yêu cầu vẽ flowchart chuẩn (có decision, process, flowline).

## IV. YÊU CẦU BÀI NỘP

1. Source code đầy đủ theo cấu trúc thư mục trên.
2. README.md giải thích cách chạy chương trình.
3. Tài liệu thiết kế (project_design.md):
    - Mô tả project.
    - Phân tích thiết kế.
    - Cách sử dụng bitmask, function pointer, setjmp.
    - Flowchart chương trình.