Hệ thống tưới cây tự động thông minh (Smart Plant Watering System - SPWS)

Thiết kế và lập trình một hệ thống nhúng quản lý việc tưới nước cho cây trồng. **Hệ thống phải có khả năng đọc dữ liệu cảm biến, đưa ra quyết định tưới nước dựa trên các tham số cấu hình, và thông báo trạng thái hoạt động**. Đề bài tập trung vào việc hiểu và hiện thực hóa một hệ thống tự động hóa thực tế với các yêu cầu về quản lý trạng thái, dữ liệu và tương tác người dùng.

Giả Định Phần Cứng và Tài Nguyên:

1. Vi điều khiển: Một vi điều khiển phổ biến với các tài nguyên cơ bản (GPIO, Timer, ADC, một lượng RAM/Flash đủ dùng cho ứng dụng nhỏ).
2. Cảm biến:
    a. Cảm biến độ ẩm đất: Đo phần trăm độ ẩm của đất (ví dụ: 0 - 100%).
    b. Cảm biến nhiệt độ môi trường: Đo nhiệt độ không khí xung quanh.
3. Thiết bị:
    a. Bơm nước mini: Được điều khiển để BẬT/TẮT việc tưới nước.
    b. LED trạng thái: Một LED RGB hoặc ba LED đơn sắc để hiển thị trạng thái hoạt động (ví dụ: Xanh - bình thường, Vàng - đang tưới, Đỏ - lỗi/cảnh báo).
    c. Nút nhấn: Hai nút nhấn vật lý:
        i. Nút 1: BẬT/TẮT chế độ tự động.
        ii. Nút 2: Kích hoạt tưới nước thủ công (khi hệ thống ở chế độ chờ)/
4. Giao tiếp (mô phỏng): Hàm gửi thông báo trạng thái hoặc dữ liệu quan trọng đến một giao diện người dùng (ví dụ: in ra console, hoặc gửi qua UART/mô-đun không dây).

Yêu Cầu Phần Mềm:

1. **Các Trạng Thái Hoạt Động (Enums)**

```c
// Chế độ hoạt động của hệ thống
typedef enum {
    MODE_AUTO,
    MODE_MANUAL
} SystemMode_t;

// Trạng thái của bơm nước
typedef enum {
    PUMP_OFF,
    PUMP_ON
} PumpState_t;

// Trạng thái của đèn LED
typedef enum {
    LED_NORMAL,                 // Xanh: Bình thường, chờ
    LED_WATERING,               // Vàng: Đang tưới
    LED_LOW_MOISTURE_ALERT,     // Đỏ nhấp nháy: Độ ẩm thấp cảnh báo
    LED_ERROR                   // Đỏ sáng liên tục: Lỗi
} LedState_t;
```

2. Cấu Trúc Dữ Liệu Chính (Structs)

```c
// Cấu trúc lưu trữ dữ liệu đọc từ cảm biến
typedef struct {
    float soilMoisturePercent;
    float airTemperatureCelsius;
} SensorData_t;

// Cấu trúc lưu trữ các thông số cài đặt của hệ thống
typedef struct {
    float minMoistureThreshold;     // Ngưỡng độ ẩm tối thiểu để bắt đầu tưới
    float maxMoistureThreshold;     // Ngưỡng độ ẩm tối đa để dừng tưới
    unsigned int maxWateringDuration_s; // Thời gian tưới tối đa (giây)
    unsigned int sensorReadInterval_s;  // Chu kỳ đọc cảm biến (giây)
    unsigned int manualWateringDuration_s; // Thời gian tưới thủ công (giây)
} SystemSettings_t;

// Cấu trúc quản lý trạng thái động của toàn bộ hệ thống
typedef struct {
    SystemMode_t currentMode;
    PumpState_t pumpState;
    LedState_t ledState;
    unsigned int wateringTimeCounter;   // Biến đếm thời gian đang tưới
    unsigned int sensorCheckCounter;    // Biến đếm cho chu kỳ đọc cảm biến
} SystemState_t;
```

3. Yêu Cầu Chức Năng

1. Khởi tạo hệ thống:
    a. Cấu hình GPIO cho các thiết bị.
    b. Thiết lập các giá trị mặc định cho SystemSettings_t.
    c. Đặt hệ thống vào trạng thái ban đầu: MODE_AUTO, bơm PUMP_OFF.

2. Đọc và xử lý cảm biến:
    a. Đọc giá trị cảm biến độ ẩm và nhiệt độ theo sensorReadInterval_s.
    b. Lưu giá trị vào SensorData_t. (Có thể lấy trung bình nhiều lần đọc để làm mịn dữ liệu).
3. Logic tưới tự động:
    a. Chỉ hoạt động ở MODE_AUTO.
    b. Nếu độ ẩm đất < minMoistureThreshold, bật bơm.
    c. Khi bơm đang chạy, theo dõi và tắt bơm nếu độ ẩm đất > maxMoistureThreshold HOẶC thời gian tưới > maxWateringDuration_s.
4. Xử lý nút nhấn:
    a. Nút **1:** Chuyển đổi giữa MODE_AUTO và MODE_MANUAL. Khi chuyển sang MODE_MANUAL, phải tắt bơm ngay lập tức nếu đang chạy.
    b. Nút **2:** Chỉ hoạt động ở MODE_MANUAL. Khi nhấn, bật bơm trong khoảng thời gian manualWateringDuration_s rồi tự tắt.
5. Điều khiển LED và Bơm:
    a. Viết các hàm riêng biệt TurnPumpOn(), TurnPumpOff().
    b. Luôn cập nhật LED theo ledState hiện tại của hệ thống.
6. Thông báo trạng thái:
    a. In ra console các thông tin quan trọng khi có sự kiện: thay đổi chế độ, bắt đầu/kết thúc tưới, giá trị cảm biến định kỳ.

4. Cấu Trúc Mã Nguồn

Tổ chức mã nguồn thành các module riêng biệt để dễ dàng quản lý và tái sử dụng.

- main.c: Chứa hàm main, vòng lặp chính, và logic tổng thể.
- config.h: Chứa tất cả các định nghĩa enum, struct và các hằng số cấu hình.
- spws_controller.c/.h: Chứa các hàm logic chính (ví dụ: SPWS_RunAutoMode(), SPWS_RunManualMode()).
- hal_sensors.c/.h: (Hardware Abstraction Layer) Chứa các hàm đọc giá trị từ cảm biến.
- hal_actuators.c/.h: Chứa các hàm điều khiển bơm và LED.
- hal_buttons.c/.h: Chứa các hàm kiểm tra trạng thái nút nhấn.

5. Các Bước Triển Khai

Bạn nên tiếp cận bài toán bằng cách xây dựng nền tảng dữ liệu trước, sau đó mới đến logic.

Bước 1: Tạo file config.h

Định nghĩa tất cả các enum và struct như trong mục 3.

Bước 2: Viết hàm System_Init() trong main.c

Khai báo các biến toàn cục cho trạng thái, cài đặt, và dữ liệu cảm biến.

Trong hàm Init, gán các giá trị mặc định ban đầu cho các biến này.

Bước 3: Mô phỏng các hàm HAL

Viết các hàm giả lập trong các file hal_...c. Ví dụ, ReadSoilMoisture() có thể trả về một giá trị cố định để bạn kiểm tra logic. TurnPumpOn() chỉ cần in ra "BOM BAT".

Bước 4: Xây dựng logic trong vòng lặp chính

Trong main(), sau khi gọi System_Init(), tạo một vòng lặp while(1).
Bên trong vòng lặp, gọi các hàm kiểm tra nút nhấn, thực thi logic theo chế độ, và cập nhật trạng thái.

Bước 5: Hoàn thiện và Tích hợp

Thay thế các hàm mô phỏng HAL bằng code thực tế tương tác với phần cứng.
Kiểm tra và tinh chỉnh lại các ngưỡng, thời gian cho phù hợp.