Elevator Control System Project

The goal of this project is to design an elevator control system using the STM32 Cortex-M3 board with FreeRTOS and hardware interrupts. The system includes a CLI for user interaction to control elevator operations such as floor selection, door opening, door closing, and maintenance mode activation.

Completion Work:

- Successfully initialized hardware peripherals: GPIO, USART2, and TIM2.
- CLI implemented using USART2 accepts commands (1, 2, 3, 4, o, c, M) for elevator control, door operations, and maintenance mode.
- Utilized FreeRTOS to implement three tasks: StatusWindowTask, ProcessUserInputTask, and StartDefaultTask to handle status updates, CLI commands, and emergency mode management.
- Integrated an emergency flag triggered by external input to halt elevator operations.
- Implemented logic to simulate elevator floor movements, including updating the elevator's current floor, status, and visual display on the terminal.
- Added a mode for simulating elevator maintenance, where all regular operations are disabled and require the system to be reset.

Requirements Addressed:

- Use of Cortex-M3 board:
    - The STM32 Cortex-M3 board was used as the primary platform for this project.
- Two external hardware interrupts:
    - Configured USART2 for UART communication and an onboard button as hardware interrupts.
- CLI for Host-Target communication:
    - Implemented CLI to run major elevator features, including floor selection, door operations, emergency and maintenance mode.
- Source code shall be written in C or C++:
    - Code was written in C, following coding standards taught in class. I mentioned in the first submission that I need to create a separate file for the functions and I did.
- FreeRTOS implemented with at least two tasks:
    - Created three FreeRTOS tasks for concurrent management of elevator operations.
- All inter-task and ISR-task communication and data passing via the FreeRTOS:
    - Created a communication mechanism using shared variables and FreeRTOS task scheduling to pass data between tasks and handle interrupts.