Youseph Saad, Omar Shohatee, and Mohammad Jouni

CIS450

12/12/2024

**Technical Documentation for CIS-450 Group Project**

1. **System Architecture**

**Diagram and Overview**

The system architecture is structured to ensure seamless interaction between the user interface, task control, and hardware components. Below is a simplified architectural diagram:

A screenshot of a computer program

Description automatically generated**Task Structure**

* **Light Task:** Responsible for updating the LED brightness level and the UI to reflect the current state of the lighting. This task takes input from the brightness knob and determines the visual representation of the brightness level.
* **Audio Task:** Provides voice feedback that matches the current brightness level. For example, when the brightness is set to 75%, a corresponding audio file ("brightness\_75.mp3") is played.

**Interaction**

* The **User Interface (UI)** allows the user to interact with the system by turning the knob to adjust brightness.
* The **Event Group** ensures synchronization between the light and audio tasks. Each task sets specific event bits to notify completion.
* The **Hardware Layer** includes:
  + LED display for visualizing brightness.
  + The physical knob as input.
  + Speakers for audio feedback.

1. **Concurrency Control Explanation**

**Mechanisms Used**

The system employs the following mechanisms to manage concurrency:

**Event Groups**

* The xEventGroupCreate, xEventGroupWaitBits, and xEventGroupSetBits functions are used to coordinate the light and audio tasks.
  + **xEventGroupCreate**: Creates an event group to manage shared state.
  + **xEventGroupSetBits**: Each task sets a specific bit in the event group upon completing its execution (bit 0 for the light task, bit 1 for the audio task).
  + **xEventGroupWaitBits**: Ensures both tasks complete before continuing the workflow.

**Task Scheduling**

* **xTaskCreate**: Used to create separate tasks for lighting and audio control. These tasks run independently but synchronize through the event group.

**Critical Sections**

* The eventBits shared resource is carefully managed to avoid race conditions.
  + For example, each task updates its respective bit in eventBits after completing its operation.

**Example Flow**

1. The **light task** updates the brightness level and sets bit 0 in the event group.
2. The **audio task** plays the appropriate audio feedback and sets bit 1 in the event group.
3. The system waits (xEventGroupWaitBits) for both bits to be set before proceeding to the next iteration of the timer callback.

**3. User Guide**

**Operating the Lighting Control Panel**

1. **Adjusting Brightness**:
   * Use the **knob** to adjust the brightness levels of the LED.
   * Brightness levels are in increments of 25% (0%, 25%, 50%, 75%, 100%).
2. **Understanding Visual Feedback**:
   * The LED display will visually reflect the selected brightness level.
     + Example: When set to 75%, the display shows the "75%" indicator.
3. **Understanding Voice Feedback**:
   * The system provides a voice announcement corresponding to the brightness level.
     + Example: Turning the knob to 50% triggers a voice announcement: "Brightness: 50%".
4. **Returning to the Menu**:
   * Long-press the knob to exit the current interface and return to the main menu.

**Troubleshooting**

* **No Audio Feedback**:
  + Ensure the speaker is connected properly.
  + Check the spiffs folder for the necessary audio files.
* **Unresponsive Knob**:
  + Verify the knob is properly connected to the hardware.
  + Check if the light\_xor or eventBits variables are updated correctly.

**Summary**

This documentation provides a comprehensive understanding of the system’s architecture, concurrency mechanisms, and user interactions. The system ensures responsive and synchronized lighting and audio feedback using advanced FreeRTOS features like tasks and event groups. For further details, refer to the code and repository structure in the /src folder.