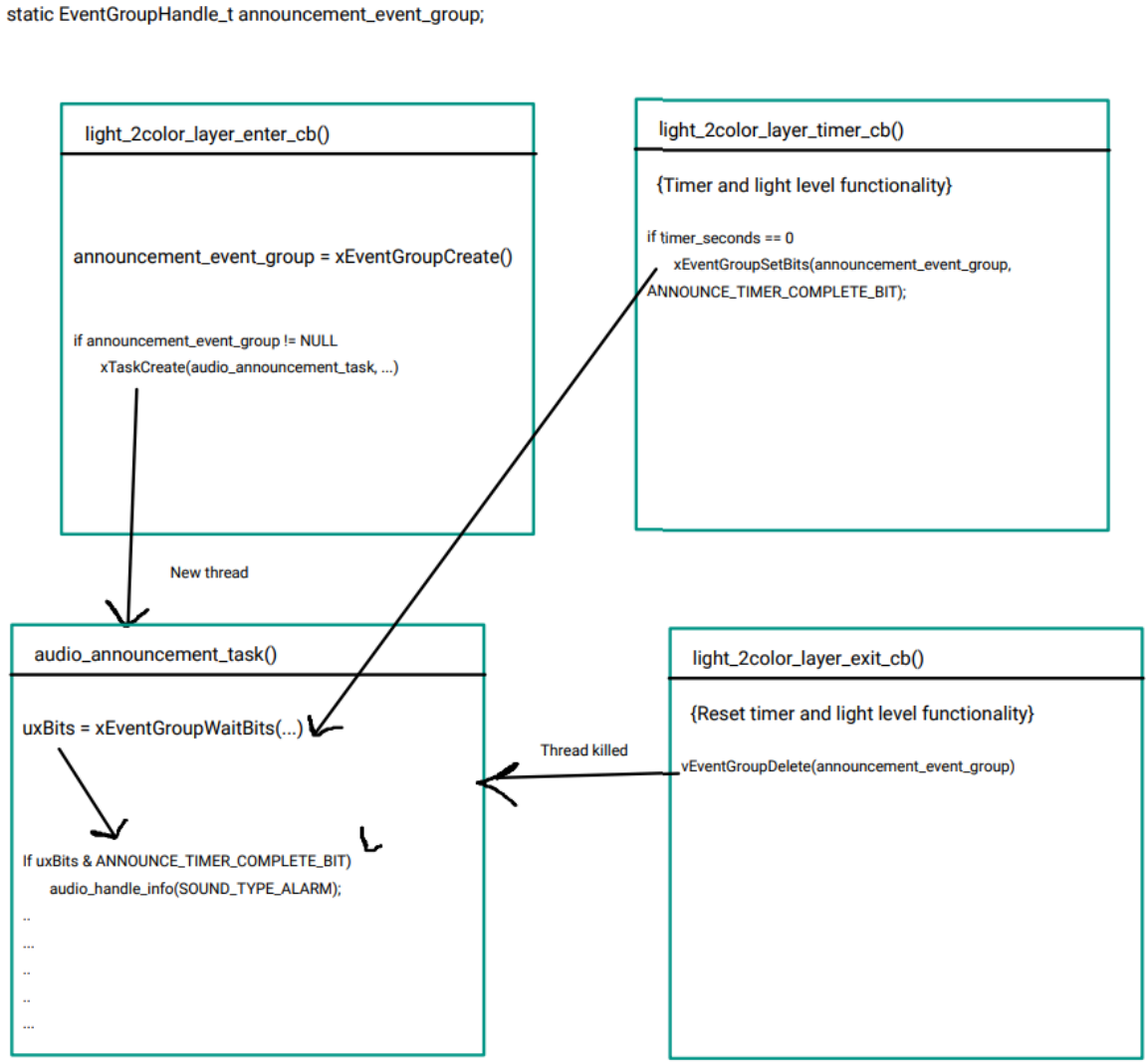
System Architecture: Diagram and description of the system's task structure and how they interact.



Above is a very simplified diagram of the system's task structure. We first begin by having the light_2color_layer_enter_cb() callback function. This function creates an event group and creates a thread called audio_announcement_task(). This thread initializes with xEventGroupWaitBits(), and is responsible for waiting for any event in which xEventGroupSet bits is called. light_2color_layer_timer_cb() is called for the bonus timer functionality. There is a light_2color_event_cb() function as well that handles the main functionality, but for simplicity it is not included in the diagram. In the timer callback, it checks the state of the timer and when it reaches 0 seconds left. Once it is 0 seconds left, it uses xEventGroupSetBits with the announcement_event_group, which will notify the audio_announcement_task, and select the appropriate audio file to play based on the wait bit. When light_2color_layer_exit_cb() callback function is called, we delete the announcement_event_group, which is turn deletes the audio_announcement_task() thread.

Concurrency Control Explanation:
Mutexes or Mutual Exclusion ensure that only one task accesses a shared resource at a time. Semaphores act like a mutex but without ownership and are often used for signaling between tasks or between an interrupt and a task. These both exist to prevent something called a race condition, a scenario in which the program accesses a variable at the same time where the outcome depends on the sequence or timing of uncontrollable events. In our source code, we use xTaskCreate, xEventGroupWaitBits, and xEventGroupSetBits to ensure safe access to shared resources. xTaskCreate creates a separate thread to prevent blocking of the main thread's execution when doing multiple operations at once. xEventGroupSetBits is a function to declare that a certain condition or state has been entered. They function similarly to counting semaphores, where each bit represents a specific event or condition. For our final, this is the lighting level value as well as other states for our bonus feature. xEventGroupWaitBits essentially acts like a listener for whenever xEventGroupSetBits is called, and when a certain state is reached it calls the proper audio handler function depending on which state it's in.

Jackson:
User Guide:

Turning the Device On

1. Turn On:
    a. When the device is plugged in via a usb-c cable it will automatically be turned on.
2. Turning Off/Resetting:
    a. To turn off or reset the device simply press the reset button located to the bottom right of the rotary knob.

Adjusting Brightness
You can easily adjust the brightness to your preferred level using the brightness adjustment knob.

1. Rotate the knob clockwise to increase brightness.
    a. The lights will become progressively brighter as you turn the knob.
2. Rotate the knob counterclockwise to decrease the brightness.
    a. The lights will dim gradually.
3. The brightness is incremented/decremented by 25% from 0% - 100%.

Understanding the Voice Announcements

- Brightness Adjustments:

- Each of the increments have their own announcement
- 25%: “25%”
- 50%: “50%”
- 75%: “75%”
- 100%: “100%”

Bonus feature implementation

The bonus feature we implemented introduces an alarm LED timer that allows users to set a timer duration. As the user turns the knob, it adjusts the timer. This functionality operates in a dedicated separate thread using the same xTaskCreate function used in the existing light level announcer functionality. The feature enhances the utility of the lighting control panel by providing a practical and interactive alarm capability that is both visually distinctive and easy to recognize. This feature allows you to adjust the LED brightness based on knob input, announces lighting levels asynchronously, and manages the alarm LED’s flashing behavior once the timer expires. It ensures the alarm task operates independently, without interfering with the primary lighting control functionality.