Project proposal

The purpose of this exercise is to create an API and an implementation to play different alarm audio sequences. In material devices, there are usually 3 different level of alarms : LOW, MEDIUM and HIGH priority.

The audio sequences are as follow:

- LOW: one beep every 30s, with a beep duration of 1s. Repeat continuously
- MEDIUM: one beep every second, with a beep duration of 250ms. Repeat continuously
- HIGH: five beeps every 500ms with duration of 250ms each, then wait for 2s with no beep. Repeat continuously

There are different rules regarding the alarms management:

- There can be only one alarm sequence at a time being played
- Only the highest priority active alarm is played at a time
- They can be started and stopped independently

If a higher priority alarm is stopped, a lower one might resume playing if still active.

What you need to do:

- You need to provide an API for the client application to start / stop alarms. The simpler, the better.
- You need to provide an appropriate architecture (classes, threads, …). It has to be robust (we're talking about alarms here), as simple as possible and readable.
- Implement the code WITH unit tests to show that the code is working.

Source code architecture shall be 'cmake' based and should allow us to run the main application as well as run the unit tests.

The application output:

- When application is executed, the output should be continuously displaying characters simulating the buzzer state. One character represents 250ms (the smaller "atomic" duration), the character '_' represents no beep (silence) and 'X' a beep (noise).
- Pressing on 'h': activates/deactivates high priority alarm
- Pressing on 'm': activates/deactivates medium priority alarm
- Pressing on 'l': activates/deactivates low priority alarm

Bonus 1: Make the program gain realtime priority and affinity to the first core (core 0) when launched (assume your program will be launched in a RT linux and that the user have the correct permission to launch it).

Bonus 2: Instead of handling only 3 different alarm levels, write your code generically so *N* priority levels are possible, with the API offering a way to create alarms of different sound patterns and with arbitrary keys for their activation/deactivation.