CODE DOCUMENTATION

1. Thread 1: Stopwatch Timer

The stopwatch timer is the most crucial thread in the program than the button monitoring thread, as it is responsible for keeping track of the elapsed time. If the stopwatch timer were to be interrupted by other threads, it could cause the timer to be inaccurate.So it is important for us to assign this thread a high priority to ensure that it runs smoothly.

Also a high priority thread means that the operating system will try to schedule it to run as often as possible, even if other threads are also waiting to run. This helps to ensure that the stopwatch timer is able to keep up with the required timing interval of 100 milliseconds. It also controls the state of two LEDs to visually indicate the stopwatch state.

2. Thread 2: Button Monitoring Thread

The button monitoring thread is responsible for watching physical buttons and responding to start/stop and reset actions. It's given a lower priority compared to the timer thread because it mainly reacts to button events and doesn't need to interfere with precise timekeeping. This priority is required because the button monitoring thread mainly checks the state of the start/stop and reset buttons.

When the start/stop button is pressed, it changes the stopwatch's state, telling us whether it's running or stopped. In case the reset button is pressed, the button monitoring thread resets the stopwatch timer, but only when the stopwatch is not in a running state.

Finally, to make sure that the most important tasks may be completed without interruption, we took into account each thread's priority inside a program. By assigning threads these appropriate priorities, we improved the overall efficiency of our program.