**Milestone 3 Lab: -.. --- - -.. --- - -.. .- … ….**

**CS-350**

Damean Murphy-Short

Southern New Hampshire University

February 3, 2025

Following completion of the Milestone 3 lab, the following questions were proposed:

**1. Why does the loop that processes the LED blinking run in a separate thread?**

There are two distinct loops of input/output that need to be accounted for: a user input loop that determines if the message should be changed or the process should be terminated and an LED blinking loop. These loops may communicate with one another through specific variables, but the intended behaviour is for an LED blinking loop to always run to completion regardless of user input. As a result, it is natural to separate the two into separate threads so that the message variables can be updated by the button thread without disrupting the behaviour of the blinking thread. This also makes the logic of the state machine model less complex because we do not need to consider a scenario where any given blinking state can be transitioned into the button process state, necessitating logic that determines which state to return to afterwards. This also avoids, as much as possible, delays in intended times between blinks.

**2. What is the purpose of returning to the off state after each completed state action?**

By requiring all states to transition from the off state instead of any other state, we can simplify our model, since most transitions only have to be accounted for when the machine is in the off state. It also ensures that the LED’s are off before transitioning to a state that will turn an LED on. The intended behaviour is to begin execution with the LED’s toggled off before enacting any action, so the off state serves as a natural starting point for each behaviour.

**3. How could you integrate serial communications to change messages available to the program?**

One can imagine a few systems that would serve to implement this behaviour. In one scenario, we could eliminate the button processing, using a messaging system instead that works as follows: wait for a message, change this full message to the active message, continue blinking this message unil there is new input (restart the loop). Likewise to the button, it would run in a thread separate from the blinking logic, as to not interrupt the blinking until a full new message is received and made active.

In another scenario, the button logic could remain, and a third thread could be added to receive messages, much like described above, except for instead of changing the activeMessage, it could append to a list of available messages. The toggleMessage() function could then be updated to select the next message from the list. It would also be possible to interpret commands from the serial communication such that messages can be removed by index in the list. The best way to do so, as far as one might think, would be to create a packet with a command flag, wherein one command would be to write a new message, followed by the message data, and another command would be to delete items from the list with integer data to specify the index.

**4. How can the LED display be used to provide debugging information?**

By utilizing the defined updateScreen() function, all of the debugging information that is typically provided on the terminal can be output on the display instead.