Jessica Ramirez

CS350

Professor Morales

5-1 Milestone Three: Input With Buttons Lab – Lab Questions

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

The loop responsible for blinking the LED must run in a separate thread so that it can operate independently of the main program logic. If the LED blinking process were placed directly in the main thread, it would block the execution of other tasks—such as user input, serial communication, or button state checking—until the blinking completed. By placing the LED logic in a separate thread, the system can continue to respond to inputs and handle events concurrently, ensuring smooth and responsive operation.

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

Returning to the off state after each completed action ensures that the system behaves predictably and does not remain in a transitional or undefined state. In the context of a state machine, the off state serves as a stable baseline from which new actions can be initiated. It also prevents unintentional repetition of actions or LED behavior, especially if button inputs or messages are being evaluated in a loop. This approach improves system clarity, reliability, and allows for better control of the LED's state transitions.

3. How could you integrate serial communications to facilitate changing the messages available to the program?

Serial communication can be integrated by using a serial interface (e.g., USB-to-TTL cable or UART) to send and receive data between the Raspberry Pi and a connected device such as a computer. By listening for incoming messages via the serial port, the program could dynamically update the active message displayed or transmitted in Morse code. This could be done by parsing user commands or text sent over the serial connection and updating the message variable accordingly. This makes the system interactive and allows real-time control without needing to stop or reprogram the device.

4. How could you use the 16x2 display to provide debugging information to the user when they don’t have access to the application console?

The 16x2 LCD display can be used to show key information about the program’s current state, such as the active message being transmitted, the current Morse code character, or error messages. For example, the display could show “Waiting for Input” when idle, or “Sending: SOS” when actively blinking the LED. Additionally, any unexpected behavior—such as invalid serial input or hardware issues—can be displayed as warnings or status updates. This is especially helpful in headless or embedded environments where access to the system console is limited or unavailable.