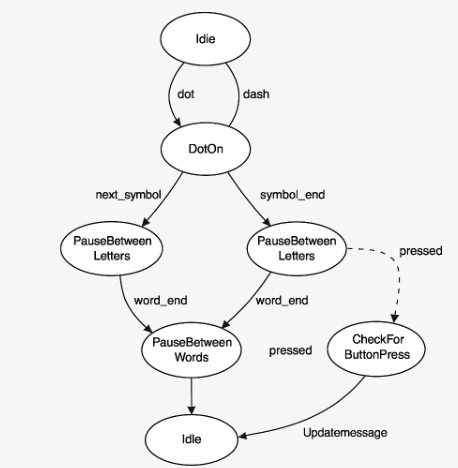
**Merrik Wright**   
**CS350**

Sorry for the late turn in on this one! I actually got back a few days ago. I was just not able to get my board to work again, so I borrowed one from my buddy, and it worked like a charm! So, I was only able to start working last night.

**I am pasting this here, as I will attach the PDF in the assignment. I think it is important here.**



**1. Why does the loop that processes the LED blinking need to run in a separate thread?**  
 Running the LED blinking loop in a separate thread allows the main program to stay responsive to other inputs or interrupts, such as button presses. If the LED logic was blocking the main thread, the program would be stuck in the blinking routine and wouldn’t be able to register the user input to switch messages or update the LCD screen. Using threading ensures the blinking pattern operates independently while the state machine can process inputs and control logic simultaneously.

**2. What is the purpose of returning to the off state after each completed state action?**  
 Returning to the OFF state between each LED action ensures that the GPIO pins are reset and prevents overlapping signals. It provides a clean, distinguishable pause that separates dots from dashes and letters from words, which is essential for interpreting Morse code. This also makes the system more readable to human observers and maintains timing consistency in the state machine logic.

**3. How could you integrate serial communications to facilitate changing the messages available to the program?**  
 Serial communication could allow external input, like from a connected PC or another device to send new Morse code messages to the Raspberry Pi. A user could input a new message via a serial terminal, and the program would read it, decode it into a Morse pattern, and store it as the next message to display. This could be managed through a queue or buffer system that stores the latest message while the current one is still finishing.

**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 can display real-time status updates, such as the current message being transmitted, the current symbol (dot or dash), or button press acknowledgment. It could also show debug states like “Button Pressed: Changing Message” or “Thread Running: Morse Msg 1.” This would help users confirm system behavior without needing to connect a monitor or terminal session to the Raspberry Pi.