Lab 5 Questions – Andrew Lemus

1. **Why run the LED blinking loop in a separate thread?**
   1. **Non-blocking timing:** The Morse timing uses sleep() (0.25–3.0 s). If you run that in the main thread, those sleeps block everything else. A dedicated thread lets timing proceed while the rest of the app stays responsive.
   2. **Responsive inputs:** The button handler (greenButton.when\_pressed) needs to fire immediately; if the main thread is sleeping inside the transmit loop, you could miss or delay button presses.
   3. **UI/LCD updates & shutdown:** LCD updates, toggling messages, and clean shutdown signals (endTransmission) can be handled without waiting for long sleeps to finish.
   4. **Cleaner structure:** Separating “producer of blinks” from “event handling/UI” keeps concerns isolated and easier to debug.
2. **Why return to the off state after each state action?**
   1. **Legal transitions:** Events are defined as toggles: off↔dot, off↔dash, and off↔pause. If you don’t return to off, the next call (e.g., another doDot()) raises TransitionNotAllowed which is an error I got.
   2. **Clear timing boundaries:** Off → (do action + timing) → Off creates crisp edges between dots/dashes/pauses, matching Morse timing rules.
   3. **Safety/clarity:** Ensures LEDs are off between symbols and prevents accidental overlap.
   4. **Simpler logic:** Every unit of work begins and ends in a known state (off), making the loop easier to reason about.
3. **How to integrate serial communications to change messages?**
   1. Use UART (GPIO14 TXD, GPIO15 RXD) or a USB-serial adapter and pyserial. Define a tiny line-based protocol, read commands in a non-blocking loop (or another thread), validate, and update your active/alternate messages.
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?**
   1. Leverage ManagedDisplay.updateScreen() to surface the most useful live info:
      1. **State & symbol:** line1 = state, line2 = symbol/remaining
      2. **Active message & source:** Show which message is transmitting and how it changed
      3. **Health/errors:** Brief codes like ERR: BAD CHAR, UART: DISCONNECTED, BTN: DEBOUNCE
      4. **Network info (at startup):** IP address so you can SSH without a monitor
      5. **Debug mode indicator:** When DEBUG=True, prepend a \* or show DBG ON to remind you the app is chatty.