Christian Busca

4-1 Assignment: Wiring an LED Display

Instructor: Roland Morales

June 1, 2025

**Reflection**

In this lab, I expanded my embedded system project by integrating a 16x2 LCD display and a potentiometer into the existing Raspberry Pi circuit. The purpose was to elevate the system's capability beyond a single LED, offering more versatile and informative visual feedback.

As I progressed through the setup, I followed the Module Four Lab Guide closely. I began by installing essential Python packages—adafruit-blinka and adafruit-circuitpython-charlcd—using pip3, after configuring the global environment to allow system package installation. With the software environment ready, I powered down the Pi and carefully wired the LCD to the breadboard. This included GPIO connections for control and data, power routing to the 5V bus, and connecting the potentiometer to regulate screen contrast.

When testing the display, I used the provided DisplayTest.py script. The characters lit up clearly after some contrast adjustments. The sleep() function within the loop immediately stood out to me—it was clear that this command plays a crucial role by pausing the loop to allow the text to remain visible long enough to be read. Without this, the display would update too quickly, overwhelming the screen and making the output unreadable. This small detail really underscored how timing is critical in embedded interface design.

The addition of a text display itself felt like a significant upgrade from the more limited binary feedback provided by an LED. In reflecting on its importance, I realized that the reason for having a text display on an embedded device is to enable real-time communication with users. This is vital for monitoring, debugging, and interfacing with the system without needing an external display or monitor. Text displays are compact, efficient, and extremely useful in embedded environments where space and power may be limited.

Moreover, the LCD effectively serves as a visual representation of a state machine. I began to think of each message on the screen as corresponding to a different system state—whether “System Ready,” “Processing,” or “Error.” Each transition in the software could trigger a message change, making the display a useful feedback mechanism for internal logic shifts. In that way, the LCD becomes not just an output device but a reflection of the system’s behavior over time.

Once the base lab requirements were complete, I decided to push the concept further by experimenting with real-world applications. I integrated OpenAI’s GPT API and transformed the LCD into a chat-based assistant that prints an AI-generated quote every five minutes while also responding to typed user input in real time. Developing this feature involved reformatting long messages to fit the 16x2 screen, setting timed events, and managing autoscrolling. The final assistant functioned like a minimalist terminal interface connected to an intelligent backend, showcasing how embedded devices can interact with cloud-based services to deliver a more engaging experience.  
  
 Overall, this lab helped reinforce the fundamentals of embedded design while giving me the opportunity to build something innovative. I successfully added a new hardware interface, learned how to control it with Python, and even developed a prototype assistant application that combines hardware with AI. The process was both educational and creatively rewarding, and I’m looking forward to evolving the project even further in future modules.