Arduino Lab 2

1 Introduction

The second and final Arduino lab for CS-24 involved developing a "not a bomb" countdown timer using two 7-segment displays to indicate time set/remaining controlled by a rotary encoder. The user may turn the rotary encoder to incrementally select a value between 0.0 and 9.9 seconds. Pressing the encoder button envokes countdown mode in which the timer decreases by 0.1 second intervals. The countdown continues until it reaches zero or is interrupted by adjustments to the rotary encoder.

2 Setup

2.1 Required components

(1) Solderless breadboard

(1) Pushbutton

(7) LEDs

(8) Resistors

(15) Wires

(1) USB A/B Cable

(1) Arduino Uno

2.2 Pre-lab exercises

The setup included the completion of two preliminary exercises: one involving blinking an LED, and the other controlling the LED using a push button to ensure familiarity with the Arduino and components. Given the lab group's strong background in hardware interfacing, the exercises were completed with ease but provided a welcomed refresher on Arduino basics.

2.3 Lab setup

The setup for the primary lab consisted of identifying all the required components and assembling the circuit. During the assembly, a minor issue was encountered in which where the cathode and anode legs of a few LEDs were reversed, causing some LEDs to remain off. This was quickly identified by setting all the LEDs to high on setup to confirm their functionality. Additionally, a multimeter was used to verify that the circuit was fully connected and performing as expected.

3 Challenges

Overall, the lab went smoothly, with only minor syntactical and logical errors. These issues were resolved, often by making use of the serial monitor to observe and debug the program's behavior.
The most noteworthy issue occurred after the lab itself was completed, during the process of preparing the .ino file for submission. While renaming the file, it was mistakenly overwritten. As this action was performed on a Linux system from the terminal, there was no trivial way to revert the action and recover the lost content (easily).

4 Solutions

It was concluded attempting to restore the overwritten file content on the Linux system would be a dead end. Therefore, attention was focused on extracting the code from the Arduino. However, this approach was also unsuccessful, as the source code is not stored on the device, only the binary. After evaluating the possibility of using a decompiler and reverse-engineering the original program, it was concluded that the process would be more trouble than it was worth. As a result, a new version of the code was written.

5 Conclusion

Overall, the lab was successful. We developed a program that could be configured to display the names of an (imaginary) user(s) in lights. The wiring was straightforward and well-organized on a breadboard. Additionally, the Arduino serial monitor proved to greatly assist in the debugging process. Following the incident of accidentally overwriting the source code, the importance of using Git for version control was emphasized.