Mathias Eriksen
mterikse@ucsc.edu
Lab 07 : Toaster Oven
November 17 2022

Summary:

In summary, the main body of this lab consisted of implementing a state machine using a single switch statement. In order to properly implememt the required state machine, a good knowledge of how state machines work was necessary. The state machine for this lab in particular was provided in the lab doc, and implemented a simple toaster oven. We had to implement the outlined behaviour for the toaster oven, using buttons and a potentiometer on the I/O shield to generate events. A 5 Hz clock was also used to generate time events for the toaster oven. Overall, implementation of this lab meant the user could adjust modes, as well as time and temperature settings on the toaster oven, as well as start the cooking and watch the timer go down in real time.

Methods:

In this lab, I followed the lab doc in the order that I went about things, and drew on a lot of previous knowledge in order to get my implementation correct. As always, I looked through the lab doc in its entirety first, and then started by implementing one thing at a time, and making sure it worked perfectly before moving on. In this lab, my first priority was to get the update OLED function properly populating the OLED. In order to get it to work correctly, the key for me was using newlines to get the sprintf() funtion looking readable to anyone reading my code. By doing this, I was able to visualize what would print a lot better, and get the OLED looking as it should quite quickly. The main part of this code was just two big if statements, which mainly split the code into two sections, one which handles setup and related states in which the time is static, and another which handles cooking and related states, in which the time is dynamically changing. Once this function was written, and properly tested, I was ready to implement the state machine.

Implementing the state machine was quite simple for me, given my previous knowledge, especially from CSE 100. A lot of the coding in the latter end of that class is state machine based, so given a drawing of the state machine I knew exactly how to implement it using the event flags and conditional state switches. The biggest problem I really had with this lab was getting the setup state right when I was starting the state machine. Specifically, one mistake I had was using if statements instead of else if when implementing the mode selection logic. This was leading to the code activating all of the if statements one at a time, and one button press would loop through all of the modes for the oven. I fixed this by writing else if, which means that only one of the statements in that block will run.

Other aspects I thought were important in this lab were updating the OLED at the proper times, like it said in the lab doc. In order to implement this, I made sure that the updateOvenOLED function was only called when a value on the OLED had changed, so the program is entirely event driven. The only call to this function that is not event driven is when I initialize it before the while loop in the main function. This allows the screen to be on as soon as the microcontroller is programmed, and makes the user experience smoother.

I found the LED part of this lab quite fun, and I tried to implement my updateOvenLEDS helper function in as few lines as possible. I found when doing this that it was possible, but I needed some other short helper functions in order to turn off the LEDS when going to the setup state, and to reset the LEDS to all on when another cooking phase is entered. I also enjoyed implementing the extra credit, which was just an extra state that is entered when the cooking time is fully done. In order to make the display blink, I used a toggle bit, like I did in the Lab 6 extra credit, which toggles on and off at two Hz. Using this toggle bit and a two Hz counter from the interrupt, I alternate using if statements between inverted and normal displays.

Results:

In conclusion, the result from this lab was a fairly satisfying toaster oven the user can run on the OLED display on the I/O shield. I also like the fact that the main while loop of this lab is a single line which calls the state machine, and all events and cases are handled within that single state machine. In total, I probably ended up spending between 10 hours total on this lab, and I didn't have a lot of time where I stuck, I mostly knew what my next step would be. Most of my time was spent thinking about how to properly implement certain things, and organize my code in a sensible manner. I did enjoy this lab, it was easier for me than the previous one, due to my knowledge of state machines, and due to this I completed the lab on my own.

Feedback:

Overall, I enjoyed this lab and did not have any real issues with how anything was presented in the lab files. The demo video was also very useful when writing the code, since it showed exact details for the behavior we were trying to implement. For the grading, I do have trouble understanding how only 9 points, or a third of the grade is based on proper behavior of the lab. In my opinion, it should be worth more points to have the behavior working in the way it was supposed to than following specs for code

organization. If I were to suggest something that could be changed in this lab, it would be a bigger extra credit portion which is worth more points for those who want to go further in challenging themselves and learning C. The extra credit in the last lab was very satisfying and brought all of the points of the lab together, while the extra credit in this one was quite small and was only worth one point. I do believe more could be done with the extra credit.