Lab 7
Ayman Ahmed Khan

1. I worked on this project myself, including the extra credit display inverting part
2. I liked this lab, because playing around with the board is one of the best parts of these labs. Since I had been using CoolTerm and was able to get used to the new workflow of programming the UNO32 in the last lab, it was much easier to get up and going for this lab. Looking at the serial output, and doing the inputs on the board itself was pretty straightforward and useful in debugging my program. This time around, we had only 2 files rather than the 5 or 6 files we had to program in the last lab. It made consolidating all the code very easy, tracking down bugs nice (because really it could only be in 1 file rather than 5), and not having to change project configurations every time I wanted to run my lab was pretty convenient. Some confusing things that the lab manual did not make clear was that the button debouncing is already being performed for us, probably in the .o file that we added. I had implemented debouncing myself in the ISR when I first started working on this lab and was getting confused why the buttons were so unresponsive. Commenting my own debouncing code seemed to do the trick, so I wish that was just made more clear in the lab manual.
3. I read the lab manual to start, and referenced it throughout the lab. My first goal was to just get an oven to show up on screen, and once I'd done that, I started making the different menus and modes, differentiating between short and long button presses, then taking in the potentiometer input to change the time and temperature variables, then getting the actual cooking and countdown part to work. I then worked on getting the screen to invert properly when the countdown was over, then finally getting the leds to countdown along with the timer, which was relatively easy compared to the screen inverting part. In between each milestone I "playtested" my oven to find any bugs or errors and went into my code, added many serial out print statements to understand why the unintentional behavior was happening, and debugged most of the big issues before moving on. I didn't ask anyone for help in this lab, because that's what I've been doing for the rest of these labs, just working by myself, which is still working for me. While I did have a few bugs that had me hitting my head on my desk I eventually solved them myself because I assume people's implementations of their toaster program would be so wildly different that going on piazza or asking others for help would take more time explaining the problem to them than just figuring it out myself.
4. I spent around 8-10 hours working on this lab, I again utilized the 1 day extension to space out my work because although I could've sat coding for 8 hours straight I really didn't want to do that. The lab manual this time around didn't have any big errors, just some confusing wording regarding whether the cursor should return to its previous position after cooking is over (as well as the time and temperature values) or if everything should get reset to default, as opposed to when the cooking is stopped midway and everything falls back to the state right before cooking started. The video again was also very helpful in showing how the user interface should look, because the lab manual also had no pictures in it to show examples of what each of the modes should look like, or what the leds should look like at certain times, or anything like that.

My lab 7 main file ended up being over 500 lines long, so it might not be the most efficient, but it definitely works! Looking for bugs, especially in the later stages of the lab, got more tedious as the main file got longer and longer. The number of new variables and enums and #defines at the top of the file grew and grew throughout the lab, and keeping functionality in separate functions (like oven state editing components in the state machine, oled editing components in the oled updater, and flag raising in the main function) was a bit painful, but it did wonders for code organization and bug localization, most of my bugs arose because I broke the rule for a second and was trying to edit oven state in main or in the oled updater instead of the state machine. I implemented my own flag system instead of countless module/global variables so that different parts of the code could talk to each other, and having that separate scope for each function in the main file ultimately made the code cleaner and prevented a lot of unintentional bugs, so I'm glad that I stuck with it in the end. I also had many confusing errors rise because of overflow errors, a lot of "counting" variables that I had like my global timer and button tick counter and such I had initialized as uint8's because that's what so many of my integer variables were initialized as. I never stopped to think that uint8 means they only have 8 bits of precision, in other words, they only go up to 255 then overflow back to 0. When my program had trouble counting down over 51 seconds (255/5 because of the 5hz ISR), I was so lost and confused until I started printing these variables to console, and saw what was happening. I ended up changing many of the uint8 variables in my program to uint16's, and all the sudden (almost) all those weird overflow bugs went away! I also had some fun with the lab myself, creating a "warming up" screen on boot up of the program, a blinking cursor to make the ui of the oven more interactive, and a bonus "nuke" mode because I wanted to play around with the animation capabilities of the oled, and having a nuke mode in my toaster seemed too funny not to do. Check it out when you get the chance! Some pictures of the nuke mode below: