Eugene Ngo

1965514

12/4/2022

Lab 4 Report

Procedure:

Task 1a:

This task requires us to look through and complete the LCD's basic functions. Looking through the lab specs, I checked tm4c1294ncpdt.h and defined ports and variables to setup LCD_GPIOInit(). Then, to setup LCD_PrintFloat() method, I looked back to lab 3. This required just properly formatting the input float to be then used in the LCD_PrintString() method.

Task 1b:

This task requires a more proper understanding of the underlying methods of the LCD before proceeding with implementing the reading and writing of temperatures. Task 1B requires us to read through the provided code and learn the functions of LCD to then print out the temperature and clock speed as they're updated. Logically, the main function here is quite similar to the Lab 3 update temperature function where it prints temperatures using the on-board sensor, being processed via the on-board ADC. Then, the inputs from the on-board buttons were mapped to change clock speeds, similar to lab 1 where they were used to change LED change rates. Thus, in the main function the on-board buttons were mapped to different frequencies for the board clock and then that value was stored in a variable. Then, the temperature value was taken similar to Lab 3, stored to a variable. Lastly, at the end of the main function, the temperatures and the frequency are printed using the LCD_PrintString() and LCD_PrintFloat() methods with the variables that were previously set, being used to determine what is printed.

Task 1c:

This task is like task 1B, but requires a understanding of the LCD_DrawRect methods and determining the resistive touch coordinates. The drawn rectangles were used to act as buttons. I used purple and yellow buttons for the different clock speeds. This time, the trigger for different clock speeds being dependent on touch inputs, rather than on-board buttons. Luckily, the SSD2119_Touch.c has a touch_read method that can return where the touch is being registered to. Thus, if you read in the touch and subtract it by the range of the x and y of the screen, you can determine the actual location of the touch, then compare that with the proper range of the rectangles to determine if the buttons are pressed. Thus, in the main function the touch-rectangle buttons were mapped to different frequencies for the board clock and then that value was stored in a variable. Then, the temperature value was taken similar to Lab 3, stored to a variable. Lastly, at the end of the main function, the temperatures and the frequency are printed using the LCD_PrintString() and LCD_PrintFloat() methods with the variables that were previously set, being used to determine what is printed.

Task 2a:

This task was like lab 2 so a lot of the code was taken from there. I added an LCD_Pattern() function to set the LCD_display and revised sys_switch() and ped_switch() using printf() to find the appropriate value range of x and y and double check the press at the start and the end of the 2 seconds to ensure a long press. The go(), warn(), stop(), and off() are functions to change virtual LED patterns. All of the logic is the same as in lab 2.

Task 2b:

This task aims to familiarize us with free RTOs. I followed the instructions as specified in the lab spec and read through the main files. I added the necessary sub files such as PLL_header, the inits, etc. Then I worked through the file and followed the instructions which were left in comments. The first step was to complete the initialization in main and then assign priority to tasks. When implementing StartStop and Pedestrian, I realized that these are the functions to update the global variables of whether the button is pressed for longer than 2s so I implemented sys_switch and ped_switch that determine a press in the button region and let the StartStop and Pedestrian function to determine the press length using tick_time as condition. Then Control and FSM is basically the same as in task 2a, except that for state transitions we can now easily use global variables as conditions.

Results:

The overall results were as specified in the spec and each task worked as intended a shown by the submitted demonstration videos. For task 1a, the LCD screen was filled with a specified color, in my case light blue. For task 1b, the temperature and clock frequency were displayed at the top of the board and the clock frequency could be updated using the two on-board switches. For task 1c, the same thing as task 1b but two virtual buttons would also be displayed on the LCD display and could be used to change the clock frequency instead of the two on-board switches. Task 2a created the traffic light system that we made back in lab 2 but was all displayed virtually on the LCD display. All the logic is the same as lab 2 as well: By default, the system will be off. On presses longer than 2 seconds buttons will register as pressed. Once the power button is pressed, then the system will turn on and alternate between 'go' state and 'stop' state until further input. If the pedestrian button is pressed then if the system is in the 'go' state, it will transition to the 'warn' state then transition into the 'stop' state and stay there until the pedestrian button is released. If the pedestrian button is pressed and held while in the 'stop' state, then the system will stay in the stop state. Task 2b is the same thing as task 2a but was implemented using RTOs.