Ze and Caitie

CSE 474: Lab 4 Report

Part 1: Procedure

Task 1: LCD Driver

1a: Complete LCD Driver

This task was a straightforward introduction to the LCD used in conjunction with the board. It required that we download the drivers, install it properly on the computer, and use the included files to simply cause the screen to show an output we desired, in this case: a screen color fill.

Figure 1: LCD Used for Tasks was SSD2119

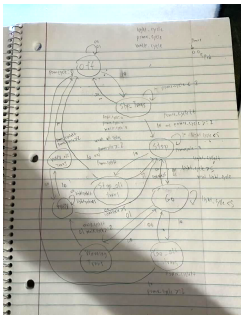
1b: Temperature Readings

The second part relied on the code and labs we did for reading the temperature of something. However, instead of displaying the temperature as raw data to the PuTTY interface we had to display it on the LCD. This required understanding how to receive data to the LCD and also display it properly. To do this, we used print statements to print the values that were being sent through without changing any other functionality of the code from Lab 3.

1c: Resistive Touch Screen

The resistive touch screen required that we know did not change the key inputs from Lab 3 so that the system clock frequency could be changed by clicking on a key of the keyboard. However, this required that virtual buttons were created and that the input could occur from clicking one of these. The resistive touch screen on the LCD then was able to receive the touch. We had to code in this implementation using the touch based LCD commands.

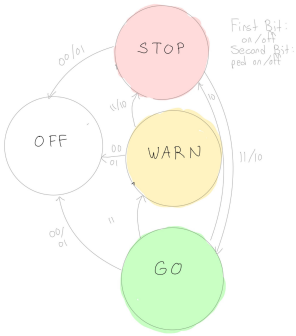
Task 2: Bare-Metal Programming vs. Real-Time Operating System



2a: Implement a virtual traffic light controller using the display module

This lab relied on Lab 2, instead of Lab 3, and it meant to build a traffic light controller but instead of using a breadboard and LEDs, we were to display the traffic light on the LCD. Therefore, we needed to make the entire design of the traffic light as well as the state machine to know what to print. We also needed to have a condition for nothing was on, to make this clear.

Figure 2: FSM for Traffic Light Controller, Task 2a (Refer to Lab 2 for Bigger Image)



2b: Implement the Traffic Light Controller using RTOS

Instead of simply printing the traffic light to the LCD board, this task required the illusion that there were multiple programs going on at once as it responded in real time. Therefore if a condition changed in the states, it would display on the LCD instantaneously.

There were three states, one for GO,WARN, and STOP. The special functionality is that the between states had their own transitions and that the WARN state was triggered by the touch of a button on the screen, much like in 1c. Also, we did not need the bluetooth

Figure 3: FSM for Traffic Light Controller, Task 2b

Part 2: Results

Task 1: LCD Driver

1a: Complete LCD Driver

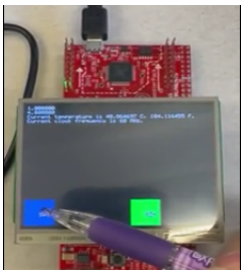
We successfully completed this method by having the LCD display red as the color of choice on the LCD. Besides the troubleshooting with downloading the drivers and reading the user manuals for the software and the hardware, there were no issues with this task.

1b: Temperature Readings

This method worked similarly to Lab 3 and we were able to print the temperature in the form of print statements to the screen. The hardest part was getting the value from the temperature sensor and saving it as a value so that it could be printed to the LCD but luckily this was able to be resolved. The method works with no issues.

Figure 4: Temperature Readings Displayed as Print Statements

1c: Resistive Touch Screen



The touch screen posed the largest of Task 1 tasks since it was the newest and most complicated of the three, since it required understanding the touch interface. Firstly, making the buttons appear on the screen was not as difficult as making these buttons have touch functionality. After the buttons were made to be able to be pressed, it was necessary to observe the change of the system clock based on the fact that the virtual button had been pressed as opposed to the keyboard button. We were able to get this completed.

Figure 5: Using Pen for Resistive Touch Screen

Task 2: Bare-Metal Programming vs. Real-Time Operating System

2a: Implement a virtual traffic light controller using the display module

We successfully displayed a virtual traffic light onto the board with the same specifications as the traffic light from Lab 2. This traffic light had the same states but rather than using the LEDs we were able to print a virtual stop light with extra design features as well. This was a straightforward task to complete.

2b: Implement the Traffic Light Controller using RTOS

This task was a lot harder to implement, but we were able to get it completed. We not only had to understand the basics and the code files of an RTOS but we also had to understand how to be guided through the steps and create a FSM after the steps had been worked through.

The initial design of the lights was encoded into the design of the stoplight itself so that it was one code block; instead of coding the stoplight display with no light emitted and then coding it entirely again when the states begun, we coded it all together. The code also responded to user interaction to trigger the pedestrian light and had transition graphics in between the lights changing colors, an added feature from 2b.

Figure 6: Used Pencil (Pictured Right) to trigger Warning Light