Lab Report 2 ("STUBBING OUT THE SYSTEM")

Authors: Yasin Alissa and Khoa Tran
ECE/CSE 474, Embedded Systems
University of Washington – Dept. of Electrical andComputer Engineering

Date: February 2, 2021

Table of Contents

1.0 Abstract

This project consists of the task of implementing a basic battery management system as it focuses on stubbing out the system. This process includes design and development of the basic system architecture, modeling the instrumentation, establishing the high-level data/control flow, managing a basic scheduler, creating a display driver, and alarm annunciation functions. Main tasks and goals are modeling the overall system architecture, coding the different modules and functions, and implementing on the ATMega board. Our final result consists of data flow between tasks that allows for battery management of a system through display inputs and outputs.

2.0 Introduction

This lab used a round robin scheduler, TCBs, inter-task communication, and general I/O to implement the first phase of a battery management system. Hardware used included the Arduino board, out touchscreen, LEDs, resistors and a switch. In terms of software, we used multiple tasks that communicated with one another using global variables to have the desired behavior.

3.0 Design Specification

The design specifications of this project includes many different aspects. To start, the main task is to design methodologies and specifications that allow for complete comprehension of the task at hand. Designing UML diagrams to model static and dynamic aspects, identify major functional blocks, and create structural diagrams that architect the system as a collection of tasks.

Besides designing the architecture of the overall system, the project consists of developing a time-based operating system, a round robin scheduler, that schedules and dispatches tasks. Additionally, the project has to implement task control blocks to perform interrelated tasks and communicate between tasks. Finally, the process of modeling and simulating the tasks and using basic input and output operations on the touch screen display and external circuits. Overall, the project consists of designing the structure and architecture of the system, and taking that information and developing task control blocks that manage tasks and data flow in order to result towards a battery management system with inputs and outputs.

4.0 Software Implementation

System block diagram depicts the implementation between the ATMega development board, a touch screen display, and external circuitry to mimic a HV battery system. The touchscreen and the ATMega are connected through the ports depicted below in the block diagram. Additionally, the block diagram demonstrates the connection between the HVIL and the contactor

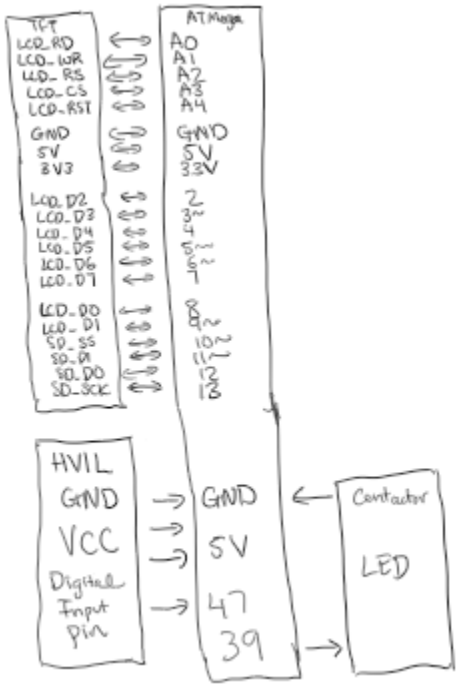

Figure 1- System Block Diagram

Structure diagram shows the event and data flog between each task function. To start off, the system controller goes towards the startup and obtains the data to transfer into the scheduler. From this point, the scheduler then loops through a series of tasks and modules, starting from measurement, SOC, contactorm, alarm, display, and lastly touchscreen. The data flows follows the chart in figure 2 beneath.

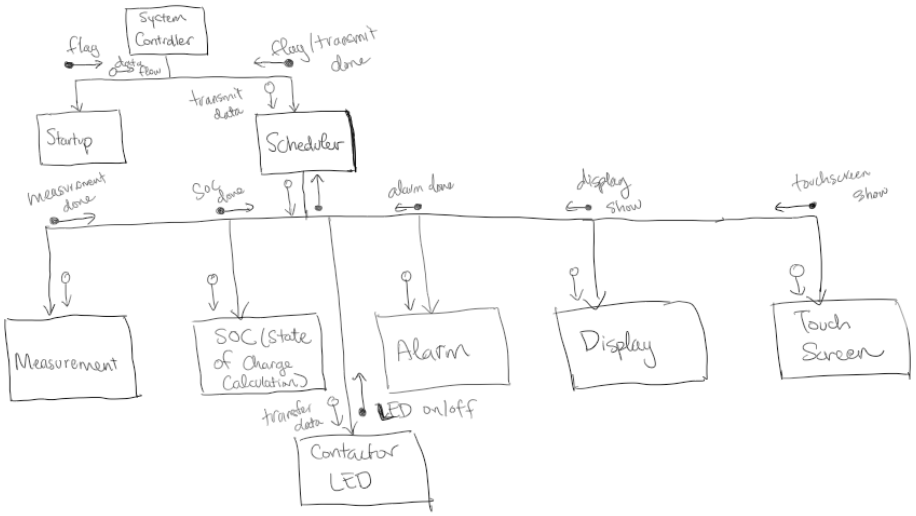

Figure 2- Structure Chart

Class Diagram goes into detail on the structure of tasks within the System Controller as reflected on the structure chart and depicts the task name, global variables that are shared between tasks, and how they are structured with each other. As depicted in figure 3, the task control block is the center of all the different task functions as it allows for communication and selection between tasks.

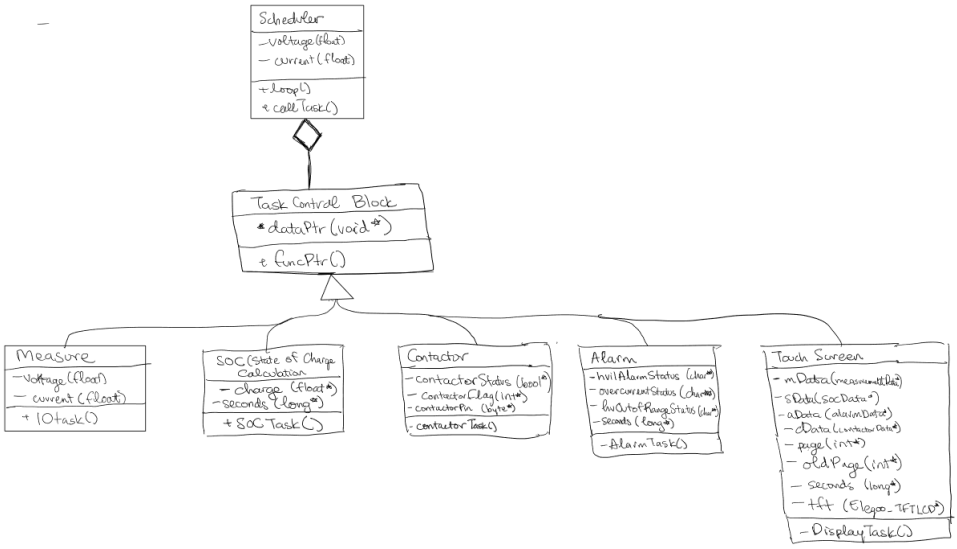

Figure 3 - Class Diagram

The data flow diagram shows the flow of data for inputs and outputs. As depicted in figure 4, the flow of data for showing measurements goes from pin 47 to the measure task, and then the display and touchscreen, keeping the voltage and current variables and values. The alarm goes through another task flow. Finally, the contactor output starts from the touchscreen and transfers the data of voltage and current into the contactor task and the port output.

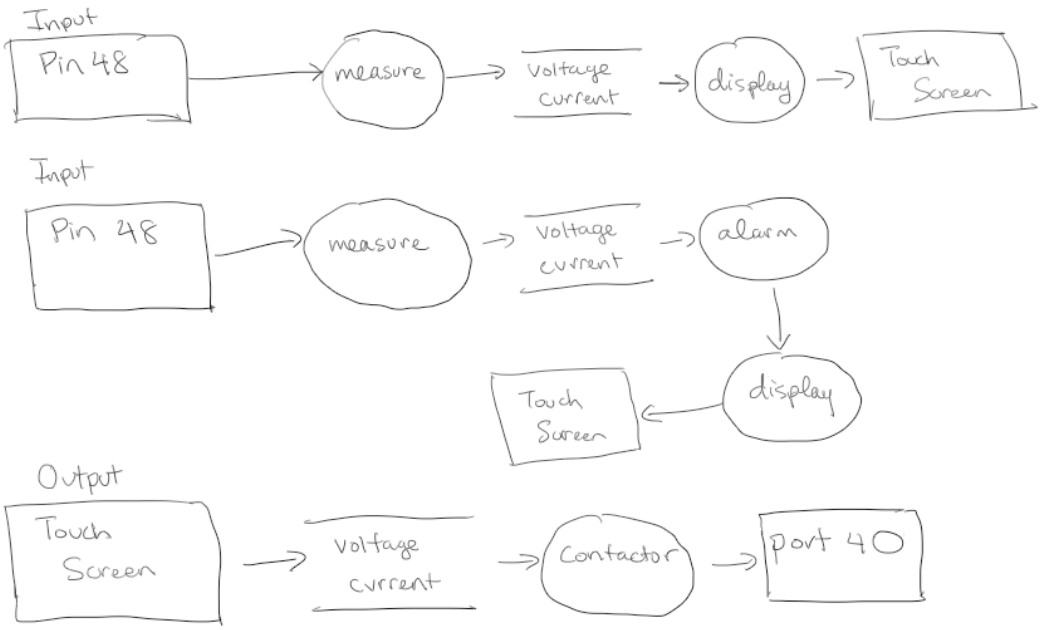

Figure 4 - Data Flow Diagrams

The activity diagram shows the system controller's dynamic behavior from initial entry into the loop function. It starts out by going into the scheduler and tasking the module to loop through a list of tasks/modules, starting from the measurement task. From the measurement task, the scheduler then calls on the SOC task, then the contactor task, then the alarm task, and the display task. Finally, it ends at the touch input task to obtain and inputs from the touch screen

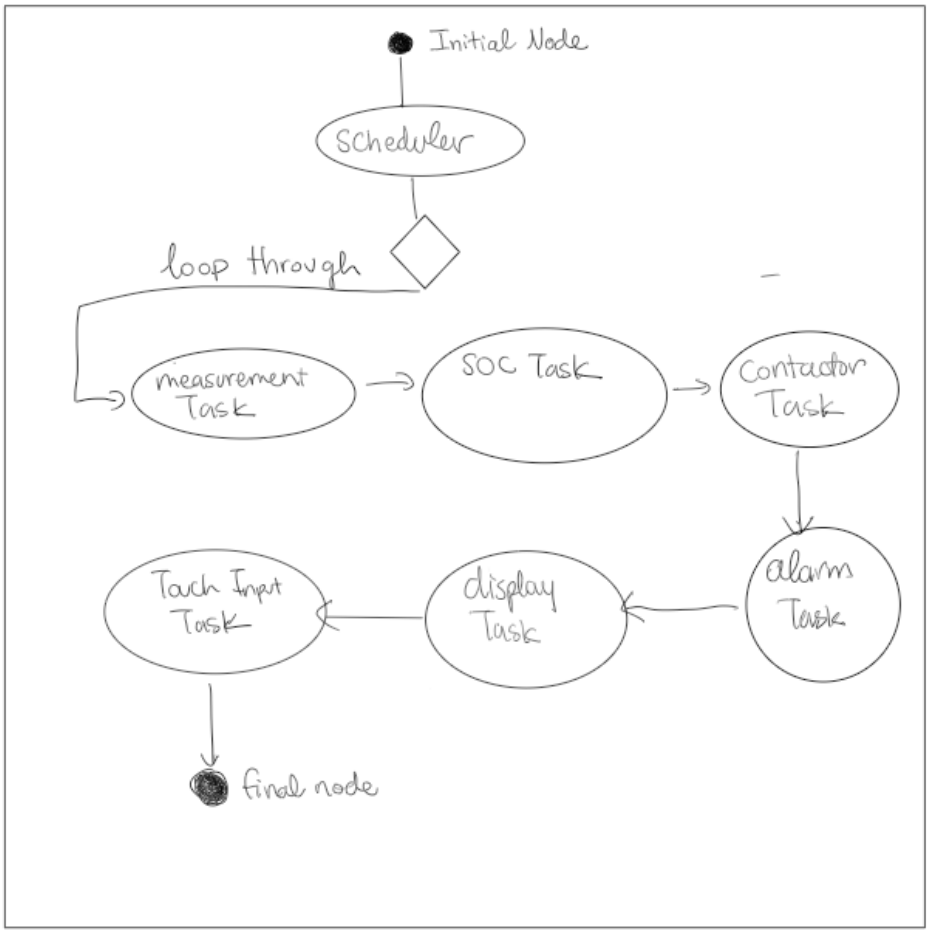

Figure 5 - Activity Diagram

Touch Screen:
Due to the complexity of the touch screen, the different sections of use case diagram, sequence diagram, and the font panel design of each screen will be shown in the diagrams. The use case diagrams show the possible user interactions with the different screens, and the possible inputs and outputs. Additionally, the interactive component is shown with a specific task.

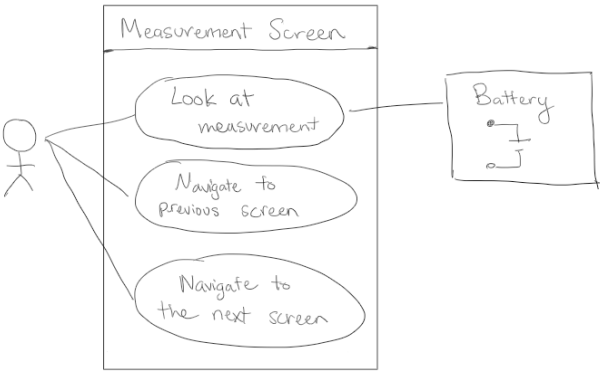

Figure 6 - Use Case Diagram for Measurement Screen

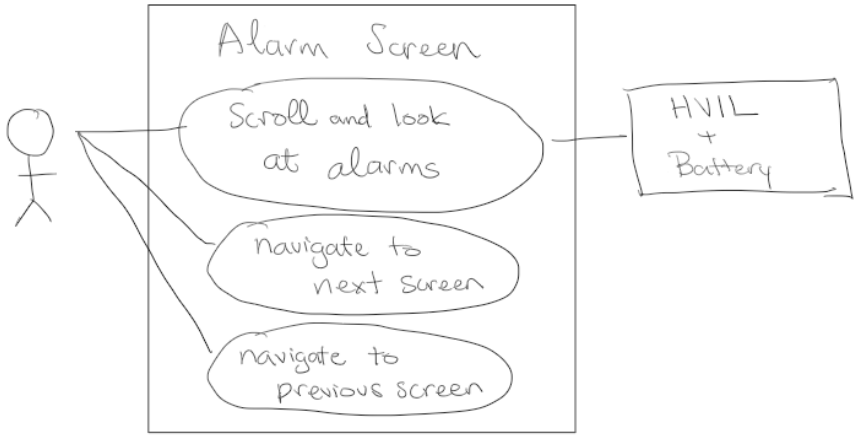

Figure 7 - Use Case Diagram for Alarm Screen

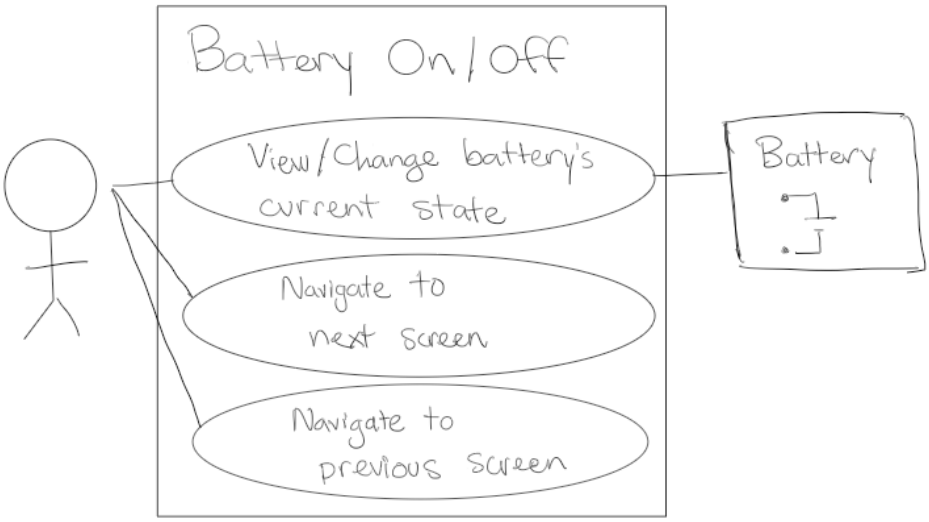

Figure 8 - Use Case Diagram for Battery Screen

The sequence diagram of each screen shows the flow of tasks in detail for each screen. As the user selects the specific screen, the tasks called will be in the order of the diagrams depicted below in figure 9, 10, and 11 for each of the individual screens.

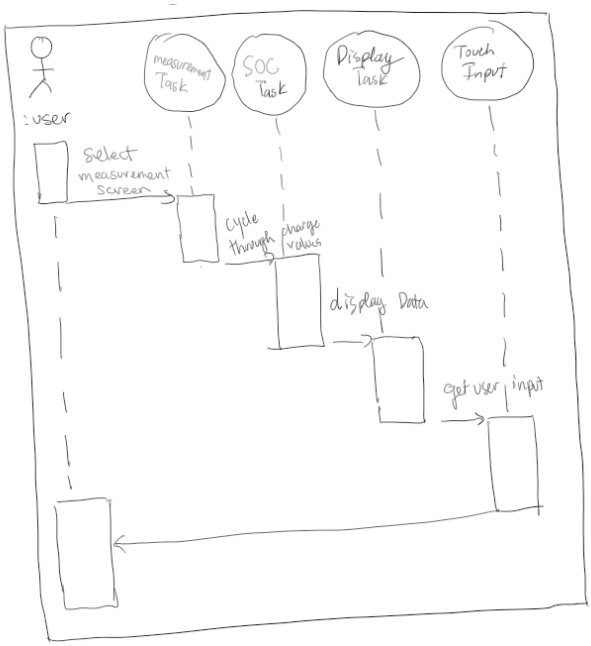

Figure 9 - Sequence Diagram for Measurement Screen

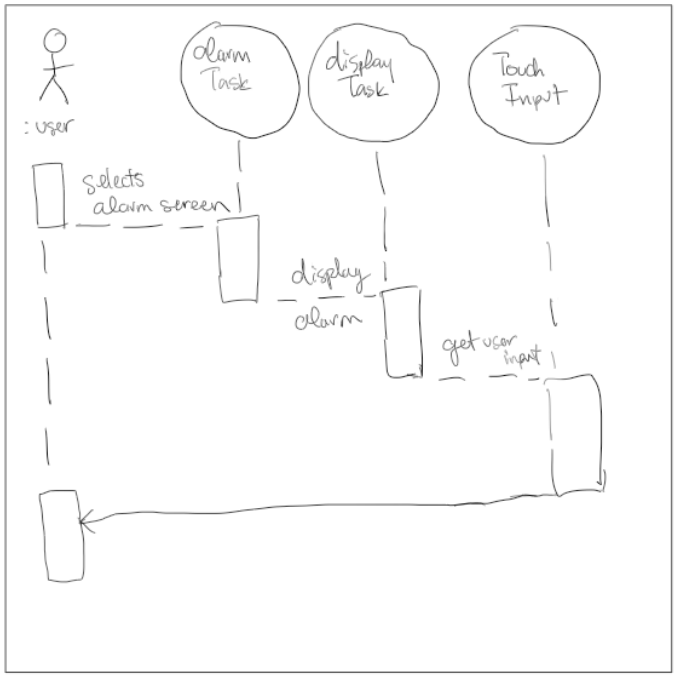

Figure 10 - Sequence Diagram for Alarm Screen

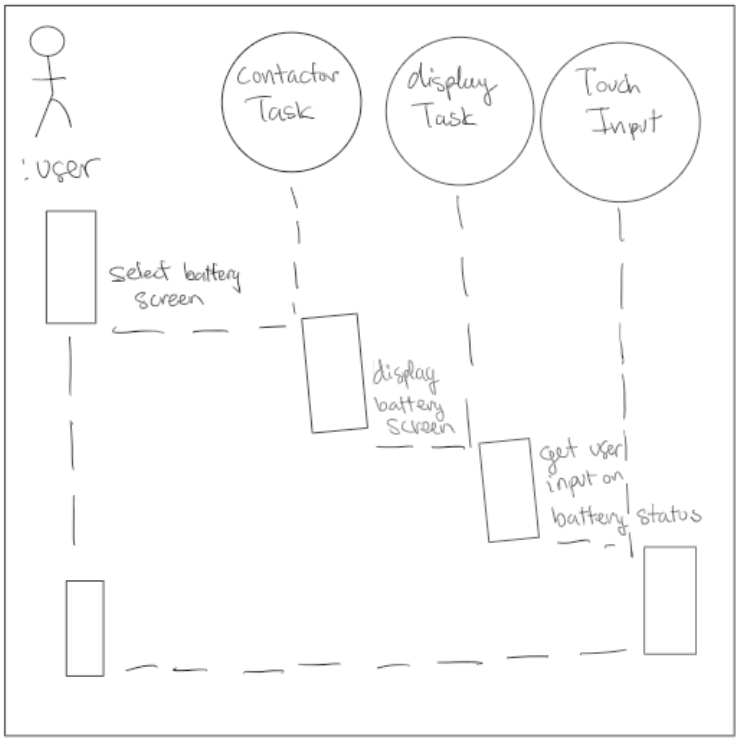

Figure 11 - Sequence Diagram for Battery Screen

Front panel design for each touch screen display shows the three different screen designs. Figure 12 shows the battery screen, which has the current battery status and the buttons to turn on and off the battery, changing the current state. The measurement screen is designed to show all the different measurements, from state of change to temperature, to the different HV components of current, voltage, and much more. Lastly, the design of the alarm screen, shown in figure 14, shows the state of each alarm. At the bottom of each screen, there are buttons to move between screens.

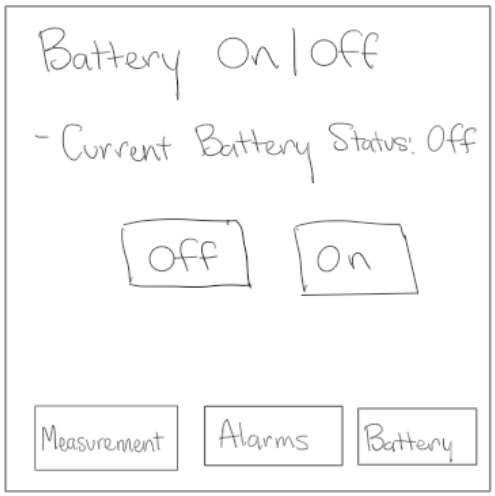

Figure 12 - Front Panel Design for Battery Screen

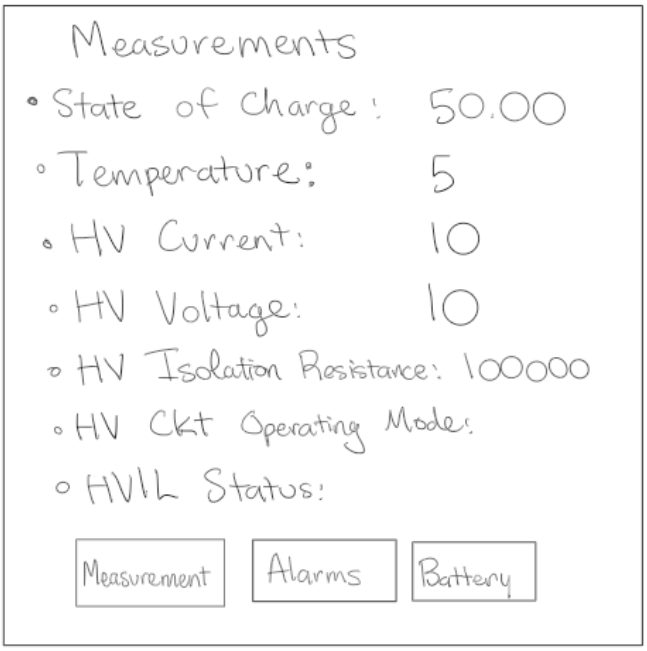

Figure 13 - Front Panel Design for Measurement Screen

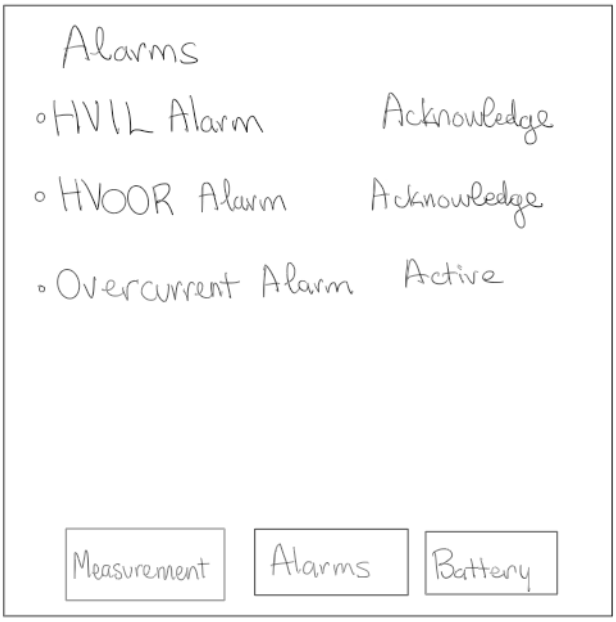

Figure 14 - Front Panel Design for Alarm Screen

State diagram for each alarm, contactor, and touch screen display shows the different states that each alarm, contactor, and touch screen moves through. In the different alarms, the difference between the state diagrams for each alarm is based on the amount of seconds that are delayed between each state. For the HVIL alarm, the delay is 1 second. For the overcurrent alarm, the delay is 2 seconds. Lastly, for the HvOutofRangeAlarm the delay is 3 seconds. This can be seen in figure 15.

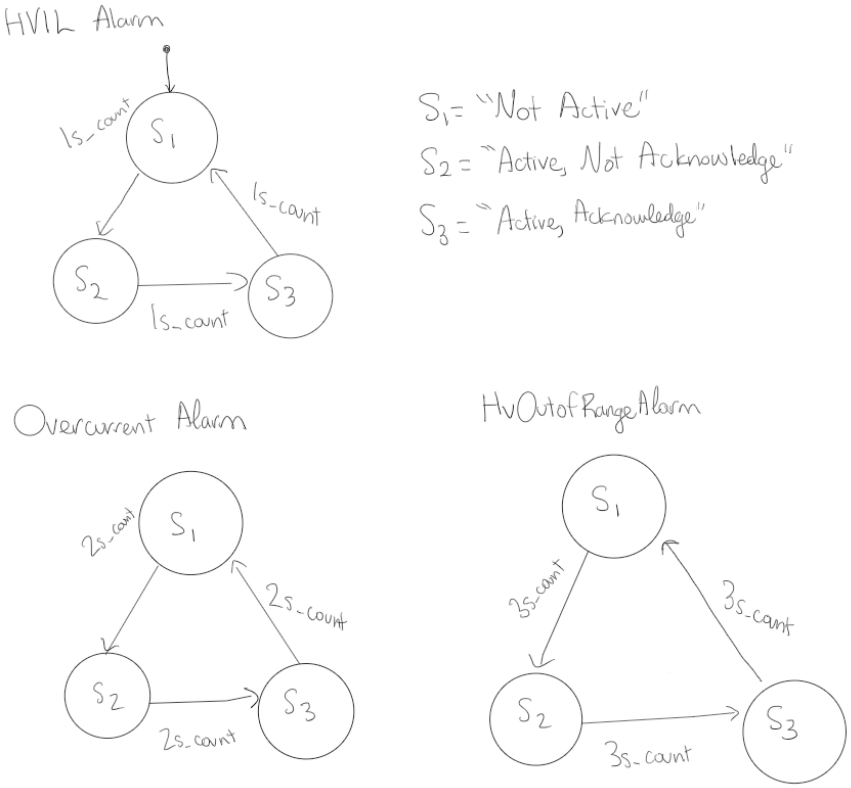

Figure 15 - State diagram for each alarm

The contactor has only two states, open and close. In close, the battery is on, while in open, the battery is off. This can be seen in figure 16.

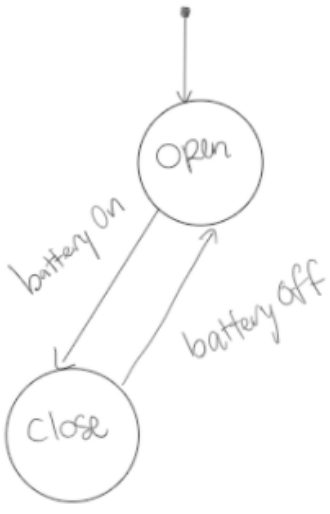

Figure 16 - State diagram for contactor

The touch screen display has three different states representing the three different screens on the touchscreen display. From the initial screen of the measurement screen, the user can select any state and traverse to the desired screen. It is the same for any other screen as the diagram shows how the user can select any screen from any given point.

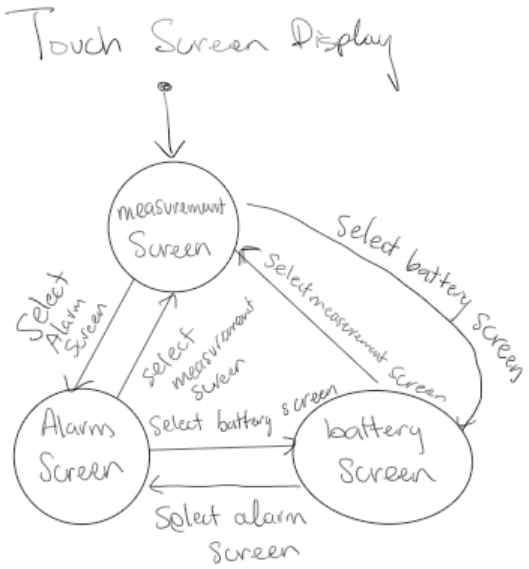

Figure 17 - State Diagram for Touch Screen Display

In conclusion, the software implementation of this project is designed from the diagrams given above and the pseudocode in the Appendix. Each task and the overall structure is depicted, showing how the entire project is developed.

5.0 Test Plan

5.1 Requirements

Scheduler Function:
This function should run each of tasks given in the TCB queue once per second.

Measurement Function:
This function should update the temperature's state once per second, the HV current's state once per 2 seconds, and the HV voltage's state once per 3 seconds. It should also change the HVIL status through user interaction.

Alarm Function:
This function should update the HVIL alarm's state once per second, the overcurrent alarm's state once per 2 seconds, and the HVOOR alarm's state once per 3 seconds. It should also change the HVIL status through user interaction.

SOC Function:
This function should update the SOC's state once per second.

Contactor Function:
This function should change the state of the contactor to open and turn off the contactor LED when the battery is turned off and change the state of the contactor to closed and turn on the contactor LED when the battery is turned on. The initial state of the contactor should be open.

Display Function:
This function should display the screen chosen by the user and show the cycling values of each aspect of the system. Values that are not changing should not be flickering.

Touch Input Function:
This function should allow the user to choose which screen they want to display through input on the touch screen. It should also allow the user to affect the contactor's state by turning the battery on and off using user input in the touch screen.

5.2 Test Coverage

Scheduler Function:
This function should run each of tasks given in the TCB queue once per second. We can test the frequency that the tasks run in to confirm they run at a 1 Hz rate

Measurement Function:
This function cycles various measurements (temperature, HV Current, HV Voltage) through set states per measurement at varying frequencies. We can test to see that the values are updating at the correct rate per measurement. This function also updates the HVIL status based on input from our circuit. We can test that the different states in our circuit lead to the correct corresponding state in the measurement function.

Alarm Function:
This function cycles various alarms (HVIL, Overcurrent, HVOOR) through set states at varying frequencies. We can test to see that the values are updating at the correct rate per alarm.

SOC Function:
This function cycles the state of charge of the systemthrough set states at a 1 Hz rate. We can test to see that the values are updating at the correct rate

Contactor Function:
This function changes the state of the contactor when the system flags it to do so. We can test to see that when this flag is to open the contactor, the contactor opens and vice versa.

Display Function:
This function displays each screen and updates the state at the correct rate for each aspect of the system. We can test that states are changing at the correct rate, only changing when they're intended to, and for the aspects of the system whose states depend on user input, we can test that user input has the desired effect on the state.

Touch Input Function:
This function changed what screen is displayed and also changed the contactor's state based on input from the user. We can test that each button is functional and has the intended effect on the state of the system and on the display itself.

5.3 Test Cases

Scheduler Function:
We tested using the millis() function and print outs to the Serial monitor how long each call to the scheduler function lasted. We found that the scheduler function runs for 1000 ms plus or minus 1 ms each time it's called, meaning that it calls each task at a 1 Hz rate.

Measurement Function:
We want to see that temperature updates every second and through timing using a stopwatch, it's fairly accurately updating once per second. We want HV current to update once per 2 seconds. We see that for every 2 times temperature updates, HV current updates once, meaning it updates once per 2 seconds as we found that temperature updates once per second. We want HV voltage to update once per 3 seconds. We see that for every 3 times

temperature updates, HV voltage updates once, meaning it updates once per 3 seconds as we found that temperature updates once per second. In order to test the HVIL status aspect of this function, we toggle the switch to see if when the LED is on, the HVIL status is closed and vice versa. We confirm that is the case, meaning that the HVIL status updates with our input.

Alarm Function:
We want to see that the HVIL alarm updates every second and through timing using a stopwatch, it's fairly accurately updating once per second. We want the overcurrent alarm to update once per 2 seconds. We see that for every 2 times the HVIL alarm updates, the overcurrent alarm updates once, meaning it updates once per 2 seconds as we found that the HVIL alarm updates once per second. We want the HVOOR alarm to update once per 3 seconds. We see that for every 3 times the HVIL alarm updates, the HVOOR alarm updates once, meaning it updates once per 3 seconds as we found that the HVIL alarm updates once per second.

SOC Function:
We want to see that the SOC updates every second and through timing using a stopwatch, it's fairly accurately updating once per second.

Contactor Function:
We want to see that when the contactor is initially open and by resetting the system, we can confirm this to be true. We also want to change the contactor's status to closed by turning the battery on through user input. When we touch the battery on button, we observe that the contactor's state changes to closed and the contactor LED turns on, which is the desired behavior. We also want to change the contactor's status to open by turning the battery off through user input. When we touch the battery off button, we observe that the contactor's state changes to open and the contactor LED turns off, which is the desired behavior.

Display Function:
We tested in previous tests that the display updates the values at their intended rates and that user input has the desired effect on the displayed states. We also want this function to not update the values when they don't need to be, such as when the HVIL hasn't been toggled. We can see that blink when they aren't being updated, showing that we are observing the intended behavior.

Touch Input Function:

We want this function to be able to transition to different pages of the display. We test this by trying to press each button and seeing if the display then shows the correct page. We tested the measure, alarm, and battery buttons and they all transitioned the display to their respective screens. Another test we need to see is if the battery on/off buttons on the battery screen have their intended actions. We see, like in our tests for the contactor function, that the battery on button closes the contactor and the battery off button opens the contactor as intended.

6.0 Presentation, Discussion, and Analysis of the Results

All modules and tasks were completed as it passes all the test cases required to pass. For each functionality of the project, the tasks executed without any errors and resulted in flawless data flow and intertask communication. The display functionality updates the values at the intended rates for each of the functions tasked in this project. The measurement screen shows the values at the correct timings as well as the alarm screen. Additionally, the connection between the touch screen display and the contactor worked seamlessly as inputs on the touch screen resulted in state changes on the battery functionality. Overall, we were successful in developing a battery management system that implements a round robin scheduler, TCBs, inter-task communication, and general I/O.

6.1 Analysis of Any Resolved Errors

A major issue we encountered was the touchscreen's inability to read when we pressed on it, even with a high pressure. Troubleshooting included rewriting the setup task from scratch and testing to see when the touchscreen stopped reading our physical input. We realized there was an issue with the location of our HVIL pin and our contactor pin. For unknown reasons, when our pins were even numbered digital pins, the touchscreen wouldn't take any touch input. Once these pins were changed to odd numbered pins, our touchscreen's input task ran perfectly normal.

6.2 Analysis of Any Unresolved Errors

We had no unresolved errors

7.0 QUESTIONS

• Execution time of each task

Used millis() function to find how many ms each task took to execute

- Measurement Task: average of 0 ms
- SOC Task: average of 0 ms
- Contactor Task: average of 0 ms
- Alarm Task: average of 0 ms
- Display Task: average of 77 ms
- Touch Input Task: average of 1 ms

• What delay is required for a 1Hz cycle time?

A total delay of 1000 ms, or 1 second, is needed to achieve a 1 Hz cycle time. Since our tasks take an average of 78 ms to run, we need an approximate delay of 992 ms.

• List of all inputs and outputs

Inputs:

- Touchscreen
- HVIL signal

Outputs:

- TFT display
- Contactor LED/status

8.0 CONCLUSION

As the purpose of this project was to develop a battery management system that implements multiple tasks and deals with inter-task communication, there was a tremendous amount of moving parts to this project. Each screen development alone was challenging in the aspect of designing the structure and flow as well as programming the different data flows between the ATMega pins, HVIL, and the contactor. However, the project was able to teach and demonstrate a huge amount of material based on designing UML diagrams and understanding how inter-task communication operates with general inputs and outputs. We don't have much recommendations or suggestions for improving this project as the topics and concepts that are covered are building blocks for more complicated projects. Overall, this project is a great introduction into programming the different inputs and outputs according to the data flow and interrelated tasks.

9.0 CONTRIBUTIONS

Both of us were able to contribute fairly to this project. We worked together in a call, dividing the work up evenly as we both wrote the programs for the applications and then wrote the lab report together. We talked about how to implement each application and then combined at the end to have one final product.

10.0 APPENDICES

Setup Task Pseudocode:

initialize the variable keeping track of how many cycles have elapsed
set up measurement task data
set up alarm task data
set up SOC task data
set up contactor task data
set up display task data
set up touch input task data
set input and output pins to respective modes

Scheduler Task Pseudocode:

save the time we started running this task
for each TCB in the TCB queue:
    perform the TCB's task
delay the system so that we exit the scheduler 1 second after we started it

Measurement Task Pseudocode:

updates HVIL state by reading from the HVIL input pin
update the temperature's value once per second
update the HV current's value once per 2 seconds
update the HV voltage's value once per 3 seconds

Alarm Task Pseudocode:

update the HVIL alarm's state once per second
update the overcurrent alarm's state once per 2 seconds
update the HVOOR alarm's state once per 3 seconds

SOC Task Pseudocode:

update the SOC's value once per second

Contactor Task Pseudocode:

```
if we need to close the contactor:
        change the contactor's status to closed
        write high voltage to the contactor pin to turn the LED on
else if we need to open the contactor:
        change the contactor's status to open
        write low voltage to the contactor pin to turn the LED off
else:
        do nothing as the contactor doesn't need to be updated
```

Display Task Pseudocode:

```
if we just started the system:
        display the buttons to change pages
if we're on the measurement page:
        if we just switched to this screen:
                display the title, bullet points, and titles for each measurement
        display the updated SOC state
        display the updated temperature state
        if we're at a number of cycles divisible by 2 or we just switched to this screen:
                display the updated HV current state
        if we're at a number of cycles divisible by 3 or we just switched to this screen:
                display the updated HV voltage state
        if the HVIL changed or we're at a new screen:
                display the updated HVIL status
else if we're on the alarms page:
        if we just switched to this screen:
                display the title, bullet points, and titles for each alarm
        display the updated HVIL alarm state
        if we're at a number of cycles divisible by 2 or we just switched to this screen:
                display the updated overcurrent alarm state
        if we're at a number of cycles divisible by 3 or we just switched to this screen:
                display the updated HVOOR alarm state
else if we're on the battery page:
        if the contactor's status has changed or we're at a new screen:
                display the updated contactor's status
update the current page we're on so the next cycle has that information
```

Touch Input Task Pseudocode:

```
get the point the user pressed on the touchscreen
if the pressure is beyond the touchscreen's threshold:
        if we pressed the measure button:
```

```
        change the page to the measurement page
if we pressed the alarm button:
        change the page to the alarm page
if we pressed the battery button:
        change the page to the battery page
if we're on the battery page:
        if we pressed the on button:
                tell the system to close the contactor
        if we pressed the off button:
                tell the system to open the contactor
```