# Lab Report 3 ("FLESHING OUT THE STUBS")

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

**Date:** **February 16, 2021**

## Table of Contents

| | | |
|---|---|---|
| 1.0 | Abstract | 3 |
| 2.0 | Introduction | 3 |
| 3.0 | Design Specification | 3-4 |
| 4.0 | Software Implementation | 5-18 |
| 5.0 | Test Plan | 19-24 |
| 5.1 | Requirements | 19-20 |
| 5.2 | Test Coverage | 20-21 |
| 5.3 | Test Cases | 22-24 |
| 6.0 | Presentation, Discussion, and Analysis of the Results | 24 |
| 6.1 | Analysis of Any Resolved Errors | 25 |
| 6.2 | Analysis of Any Unresolved Errors | 25 |
| 7.0 | Questions | 25-26 |
| 8.0 | Conclusion | 26 |
| 9.0 | Contributions | 26 |
| 10.0 | Appendices | 27-29 |

## 1.0 Abstract

This project consists of upgrading the previous battery management system as it focuses on fleshing out the stubs. This process includes design and development of the system's architecture, developing a linked list task queue for the scheduler instead of a round robin array, updating functionalities and state diagrams of the measurement, alarm, and contactor tasks. Besides these changes, working with interrupts and time base flags are a critical part to this project as the two interrupts allow creation of a real-time time base and flags for state transitions of the HVIL. Overall, 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 uses a linked list task queue instead of a round robin scheduler, TCBs, inter-task communication, and general I/O from analog sensors to upgrade and improve the first phase of the battery management system. Hardware used included the Arduino board, out touchscreen, LEDs, resistors, potentiometers, and a switch. In terms of software, we used multiple tasks that communicated with one another using global variables to have the desired behavior. Throughout our work on this lab, we encountered issues, some that we managed to resolve and some that we didn't. By collaborating and using the design specification and software implementation below, our group managed to successfully update our battery management system to use analog inputs and interrupts along with a dynamic linked list task queue dictated by a hardware timer.

## 3.0 Design Specification

The design specifications of this project includes many different aspects. The overall task is to implement modifications to the previous battery management system. To start, the main task is to design and implement a dynamic task queue, using a linked list to coordinate previous and next tasks for the scheduler. Another task is to work with internal and external interrupts and interrupt service routines to develop a hardware time base to set the execution rate for the scheduler. Besides hardware time base, interrupt should also work with the transition of the HVIL loop from closed to open. Besides these new features, this project also introduces several peripheral devices working with input and output operations to implement new features and capabilities to the system.

The design of the system is described in the UML diagrams in software implementation, but the main features are based on the measurement task taking analog measurements of HV current, HV voltage, and temperature. The touch screen is modified to have any active but not acknowledged alarm to force the state flow towards the alarm task, displaying the alarm, and allowing the user to acknowledge the alarm. Any active alarm that isn't acknowledged forces the user view the alarm screen and acknowledge given alarms. Lastly, the contactor task is to operate based on the touch input and HVIL alarm status. Overall, the project consists of designing the modified structure and architecture of the battery management system, and taking that information and developing task control blocks that manage tasks and data flow in order to result towards an improved battery management system with new inputs and outputs.

## 4.0 Software Implementation

System block diagram depicts the implementation between the ATMega development board, a touch screen display, LED contactor, and an external circuitry to mimic a HV battery system, dealing with the HV voltage, HV current, and the temperature. The touchscreen and the ATMega are connected through ports depicted in the block diagram. The contactor, HVIL, HV voltage, HV current, and the temperature sensor are connected to the ATMega through the ports given below.

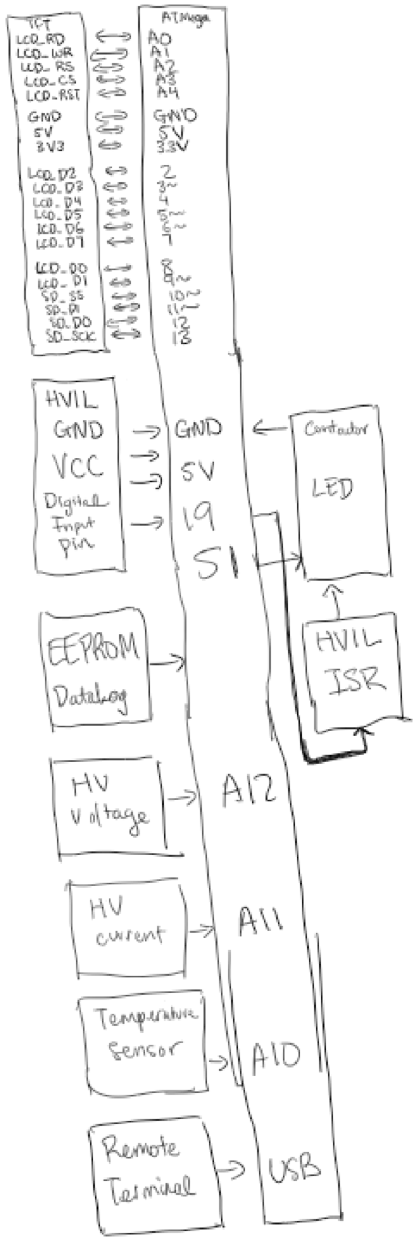


Figure 1 -- System Block Diagram

The structure diagram shows the event and data flag between each task function. To start off, the system controller goes towards the startup and obtains the data to transfer into the scheduler. The HVIL interrupt and Timer1 interrupt are initialized in the startup of the system controller, and when they run, they set a flag that signals for the main loop to cycle again. The scheduler is based on a dynamic task queue that is a linked list between the tasks. Starting from measurement, next tasks go to SOC, then contactor, then alarm, then display, and finally the 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 taskfunctions as it allows for communication and selection between tasks. The task queue, though, is a linked list between the tasks. In Figure 3 below, it indicates the starting tasks and the previous and next function of each task.

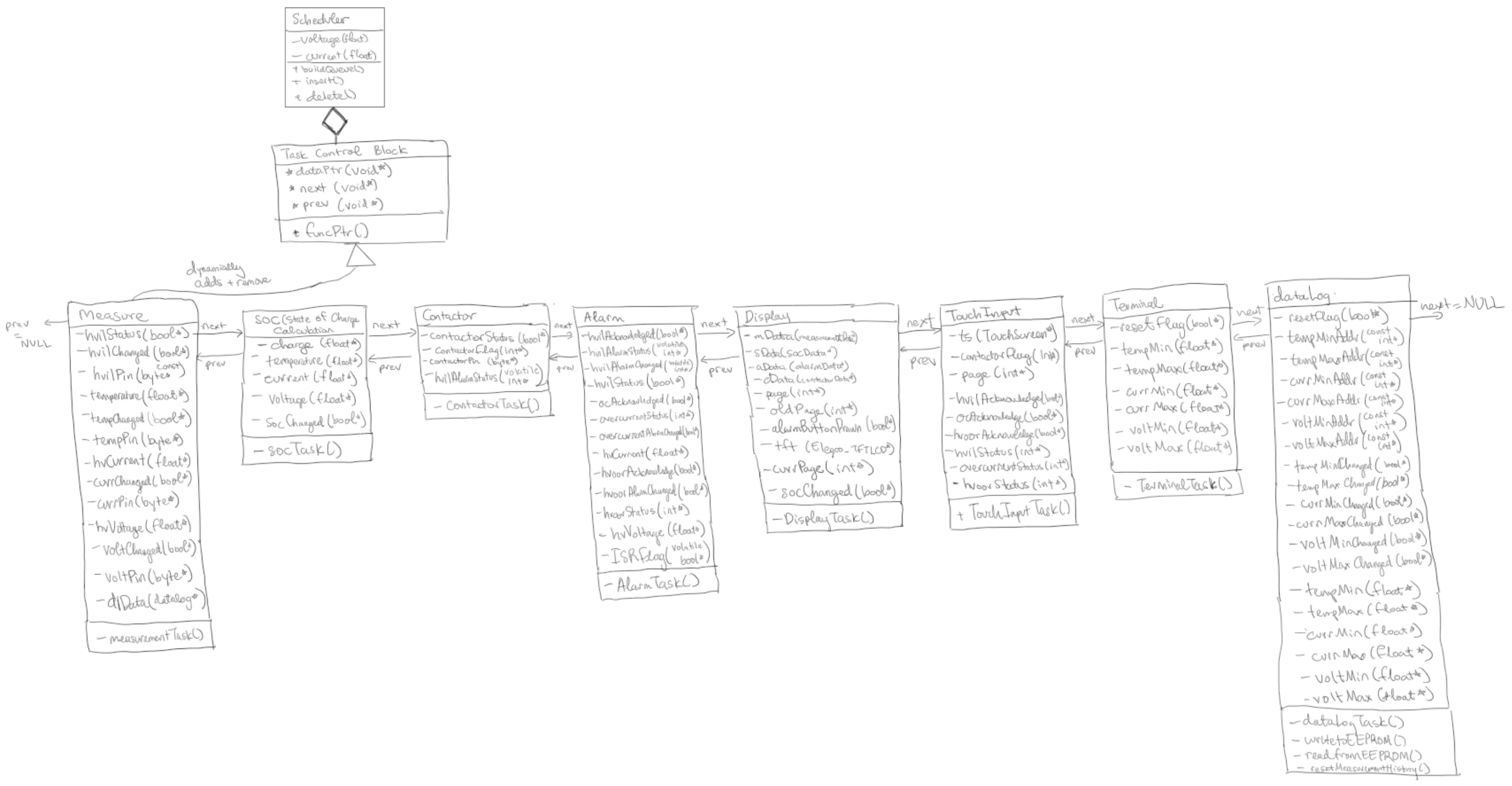


Figure 3 -- Class Diagram

The data flow diagram shows the flow of data for inputs and outputs. As depicted in Figure 4, the flow of data for the measurement tasks goes from pin 19, A10, A11, and A12 to the measure task, and then the display and touchscreen, keeping the voltage, current, and temperature variables and values. Pin 19 is for the HVIL, pin A10 is for the temperature sensor, pin A11 is for the HV current, and pin A12 is for the HV voltage. The alarm goes through another task flow, getting the input from the touchscreen to acknowledge alarms. Finally, the contactor output starts from the touchscreen and data from the alarm screen to change the contactor state in the contactor task and the port output.

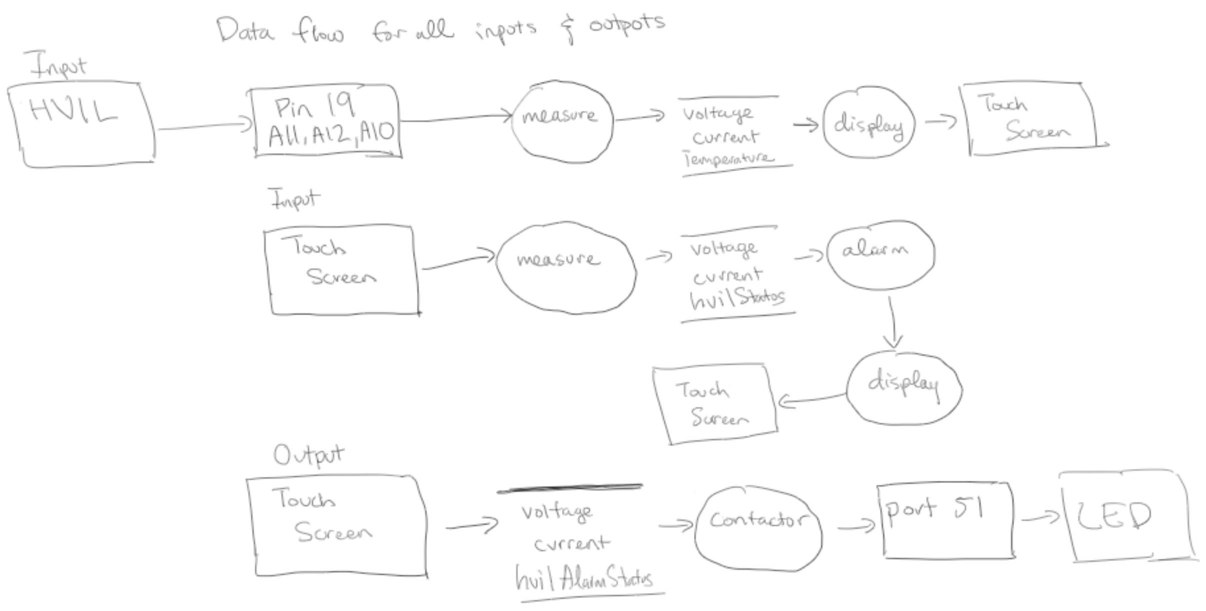
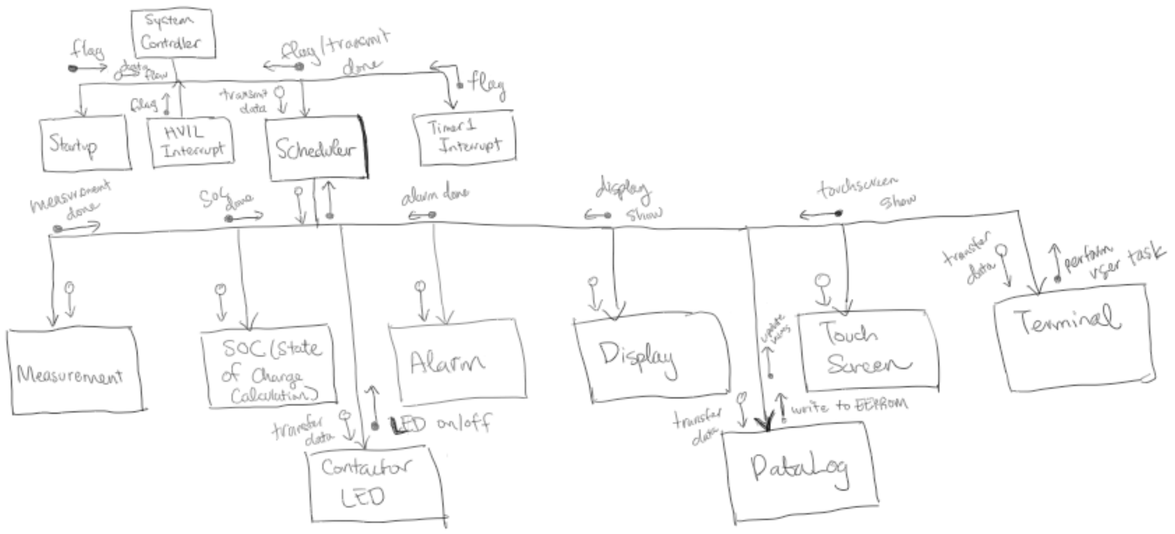


Figure 4 -- Data Flow Diagrams

The activity diagram shows the system controller's dynamic behavior from initial entry into the loop function. It starts out by going through an if/else statement to test for the time base flag. If the flag is set, then it goes into the scheduler and allows the module to loop through a linked list of tasks/modules, starting from the measurement task. The dynamic task queue allows for next and previous tasks with the linked list. 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 inputs from the touch screen.


System Controller
flag
data flow
flag / transmit done
flag
Startup
flag
HVIL Interrupt
transmit data
Scheduler
Timer1 Interrupt
measurement done
SOC done
alarm done
display show
touchscreen show
transfer data
perform user task
Measurement
SOC (State of Charge Calculation)
Alarm
Display
update vars
Touch Screen
Terminal
transfer data
LED on/off
Contactor LED
transfer data
write to EEPROM
DataLog

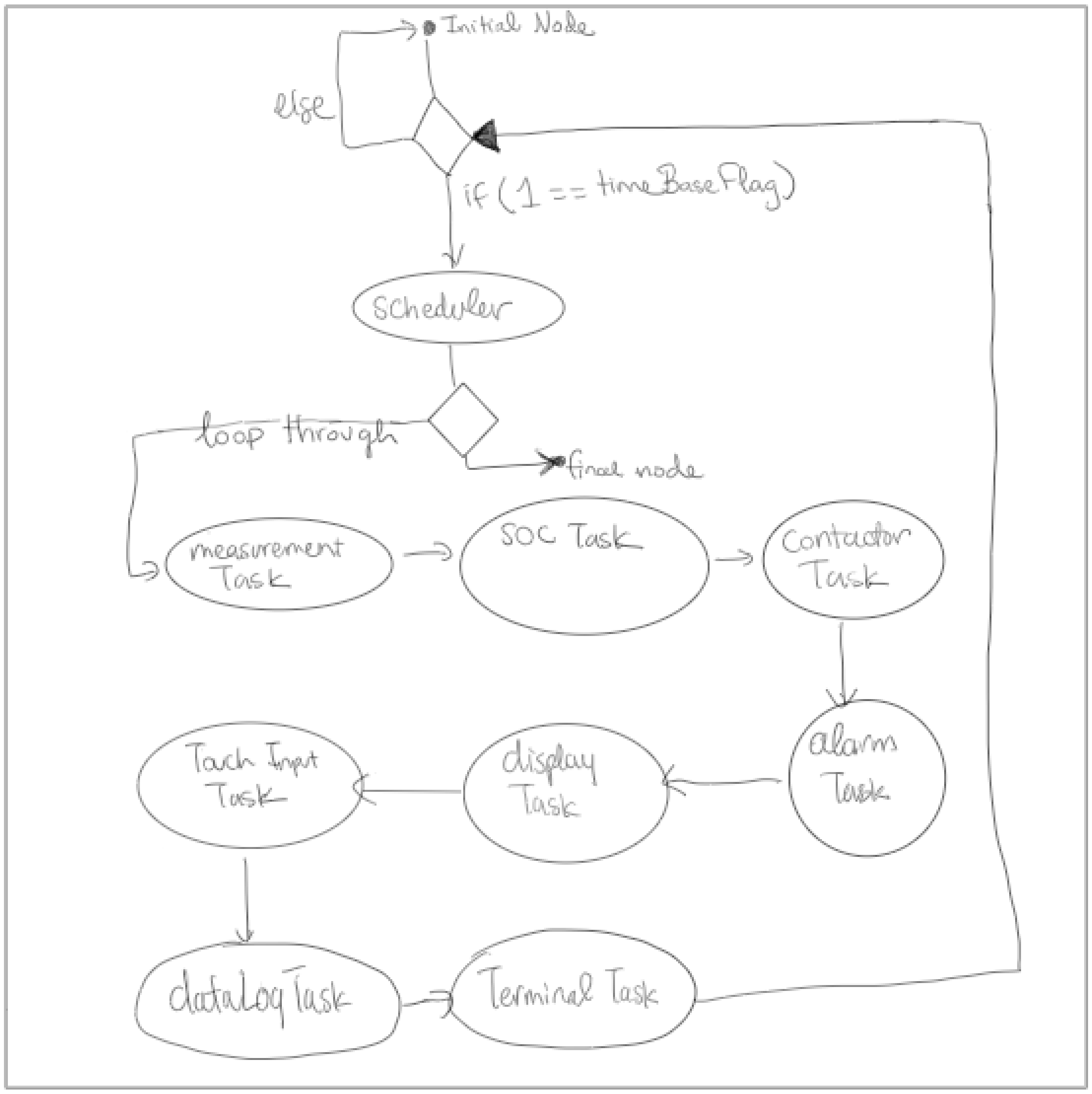


Figure 5 -- Activity Diagram

## Touch Screen:

Due to the complexity of the touch screen, the different sections of use case diagram, sequence diagram, and the front 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.

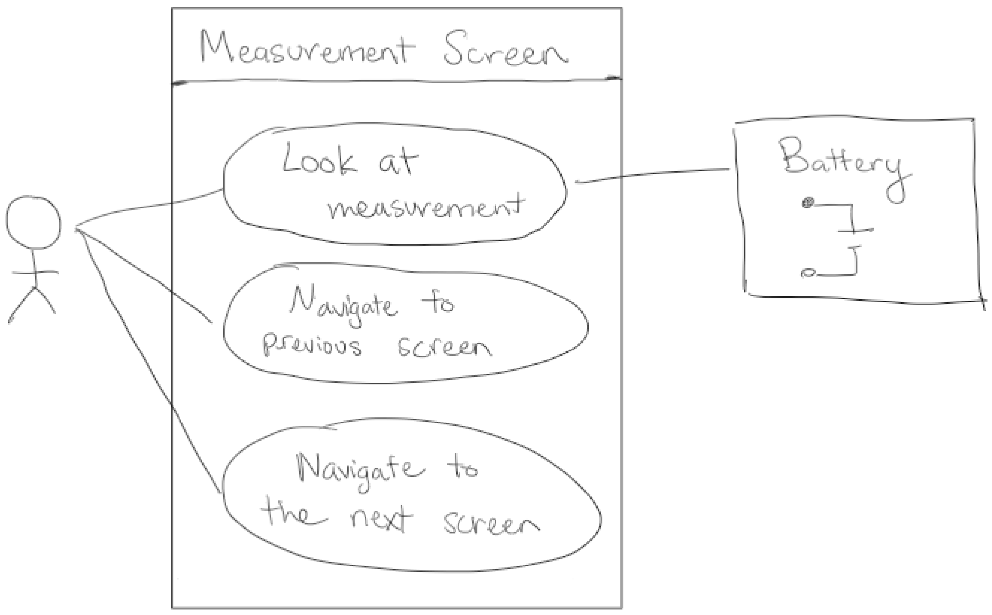


Figure 6 -- Use Case Diagram for Measurement Screen

Slightly different from the previous project, gives the user the ability to acknowledge alarms.

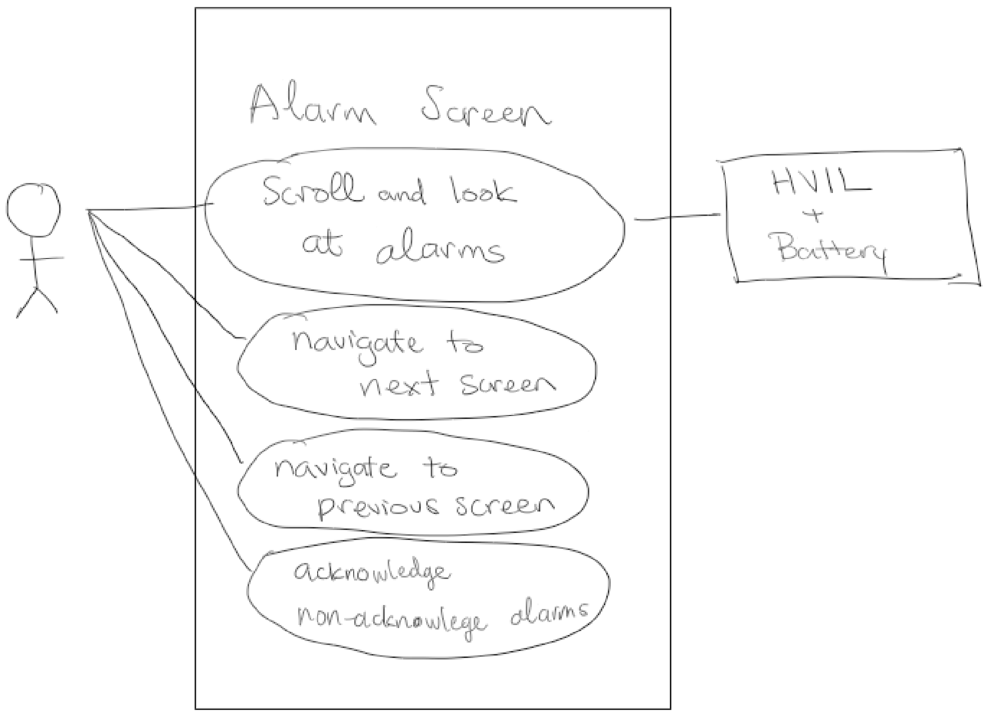

Alarm Screen
Scroll and look at alarms
navigate to next screen
navigate to previous screen
acknowledge non-acknowlege alarms
HVIL + Battery

**Figure 7 -- Use Case Diagram for Alarm Screen**

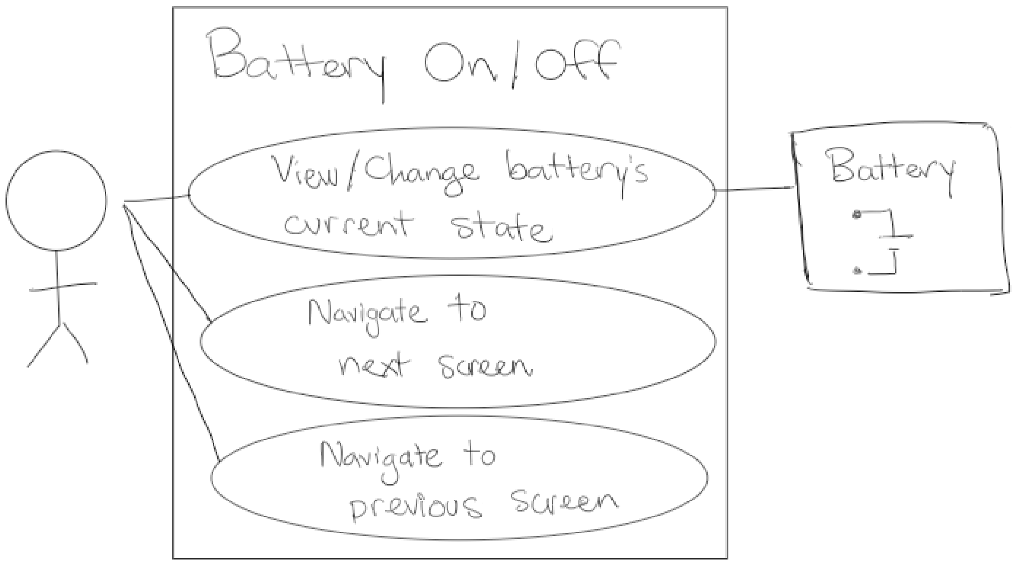

Battery On/off
View/Change battery's current state
Navigate to next screen
Navigate to previous screen
Battery

**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 Figures 9, 10, and 11 for each of the individualscreens. The individual tasks for each diagram are different from the previous project as it looks for state changes in the alarm screen and the battery screen.

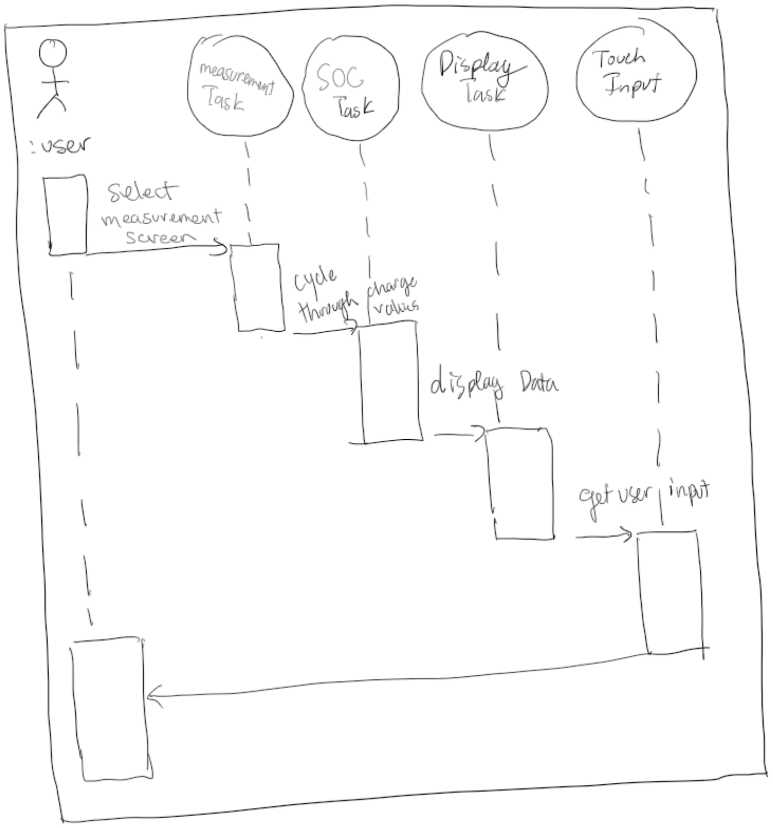


Figure 9 -- Sequence Diagram for Measurement Screen

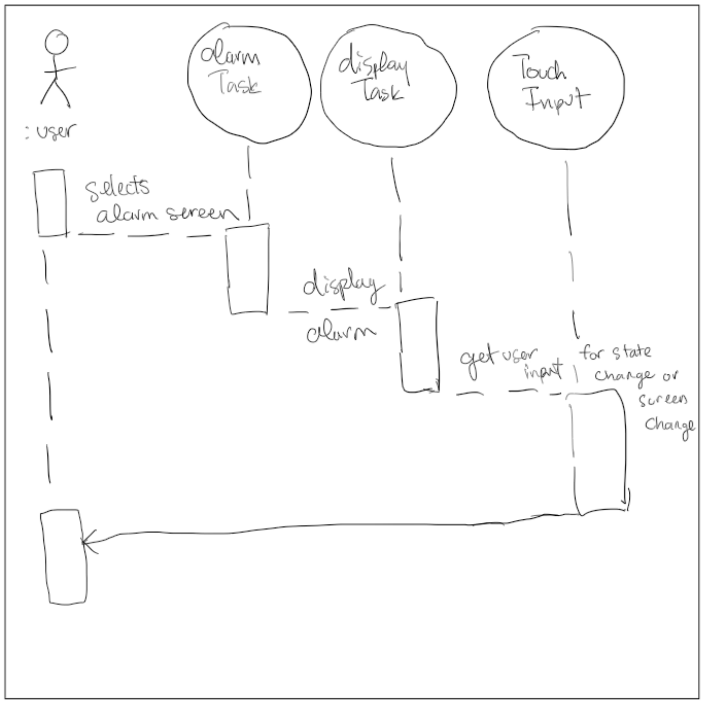


Figure 10 -- Sequence Diagram for Alarm Screen

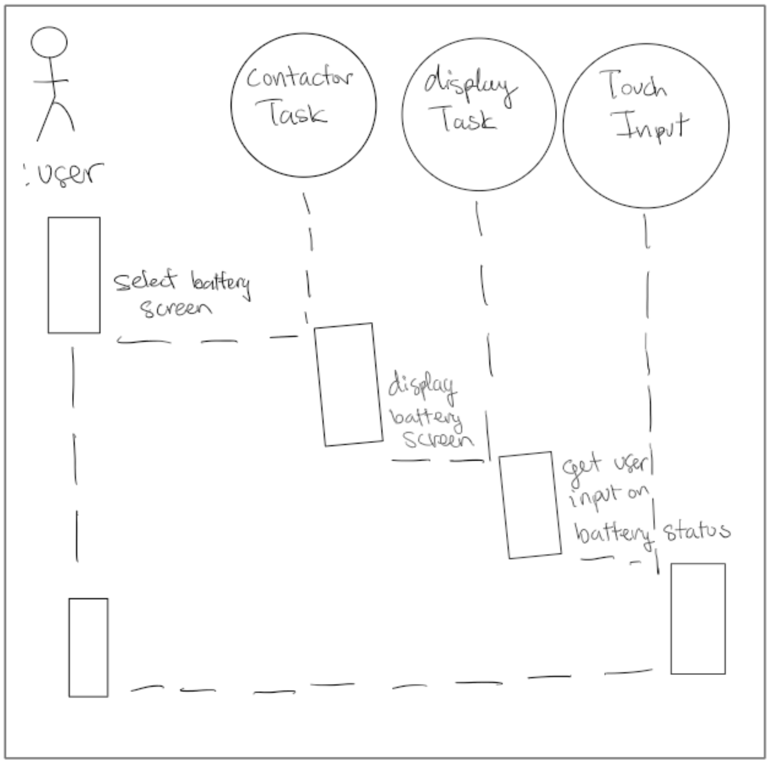


Figure 11 -- Sequence Diagram for Battery Screen

Front panel design for each touch screen displays 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 currentstate. 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 the HVIL status. Lastly, the design of the alarm screen, shown in Figure 14, shows the state of eachof the three alarms. At the bottom of the alarm screen is a section that will display a button with the ability to acknowledge alarms. As a result, the screen will only show the button if an alarm is active but not acknowledged. Whenever this appears, the touch screen automatically goes to the alarm screen to notify the user to acknowledge a recently new active alarm. Lastly, at the bottom of each screen, there are buttons to move between screens.

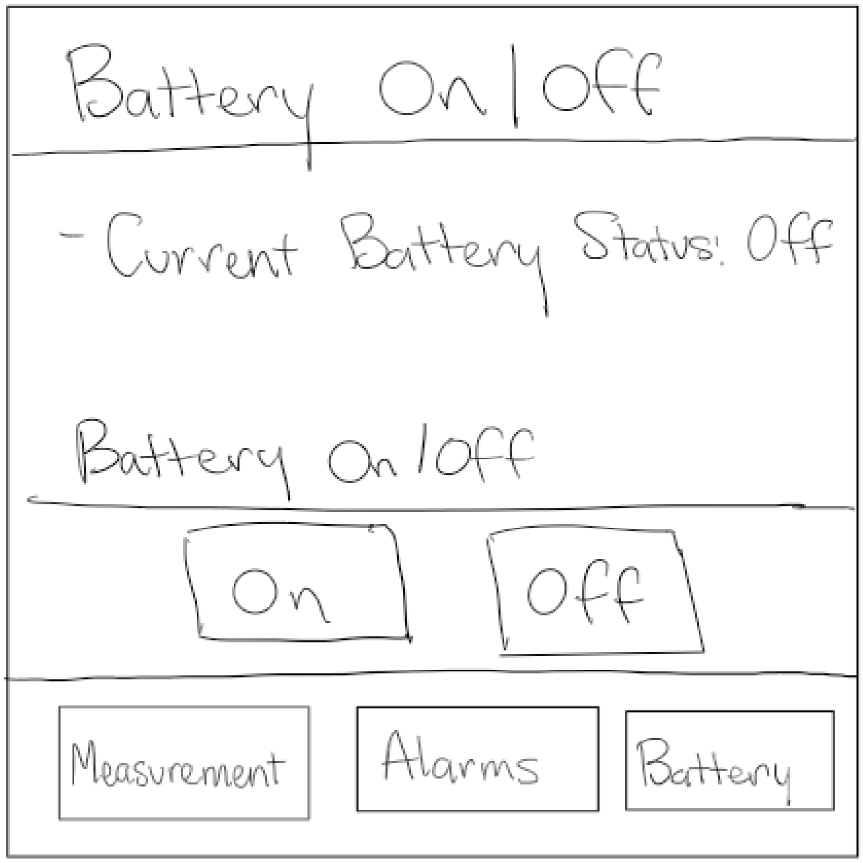


Figure 12 -- Front Panel Design for Battery Screen

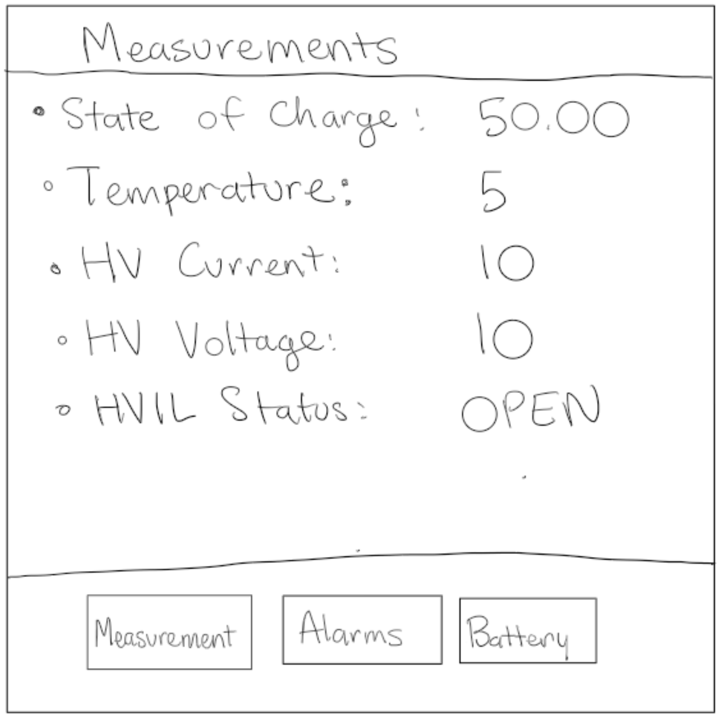


**Figure 13 -- Front Panel Design for Measurement Screen**

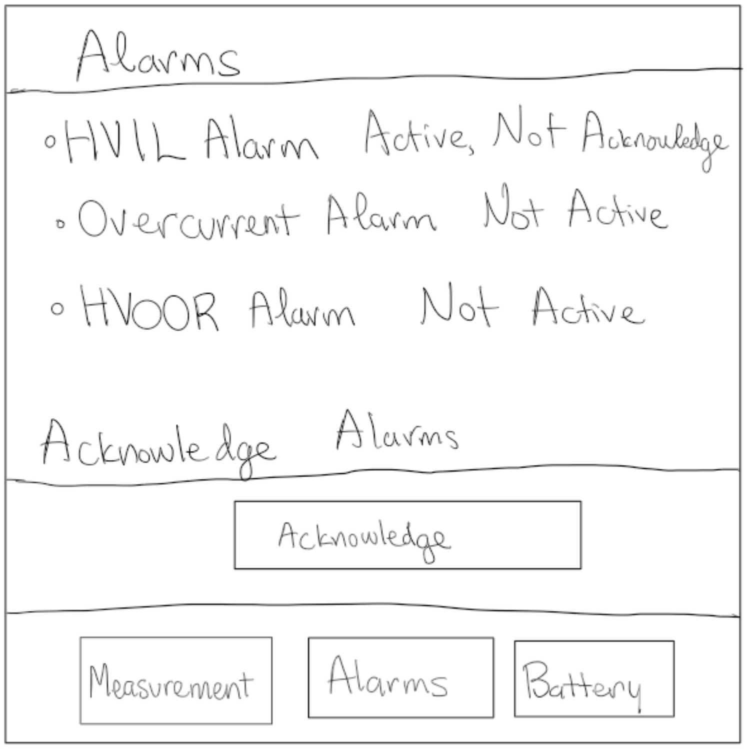


**Figure 14 -- Front Panel Design for Alarm Screen**

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 HVIL alarm, Figure 15 top, the transition between state 1 (S1) and state 2 (S2) is based on the HVIL status of either open or close. If the status is open, it would transition from state 1 to state 2 and vice versa if the status is closed. For the transition to state 3 (S3) from state 2 (S2), the input is based on the touch input from the user, acknowledging the alarm. For all the three different alarms, the transition from state 2 to state 3 is the same. However, the transition between state 1 and state 2 and state 3 back to state 1 are different. In Figure 15, it states the differences between the transition for each alarm, either being in open or close status, or the value is in either in or outside of the desired range.

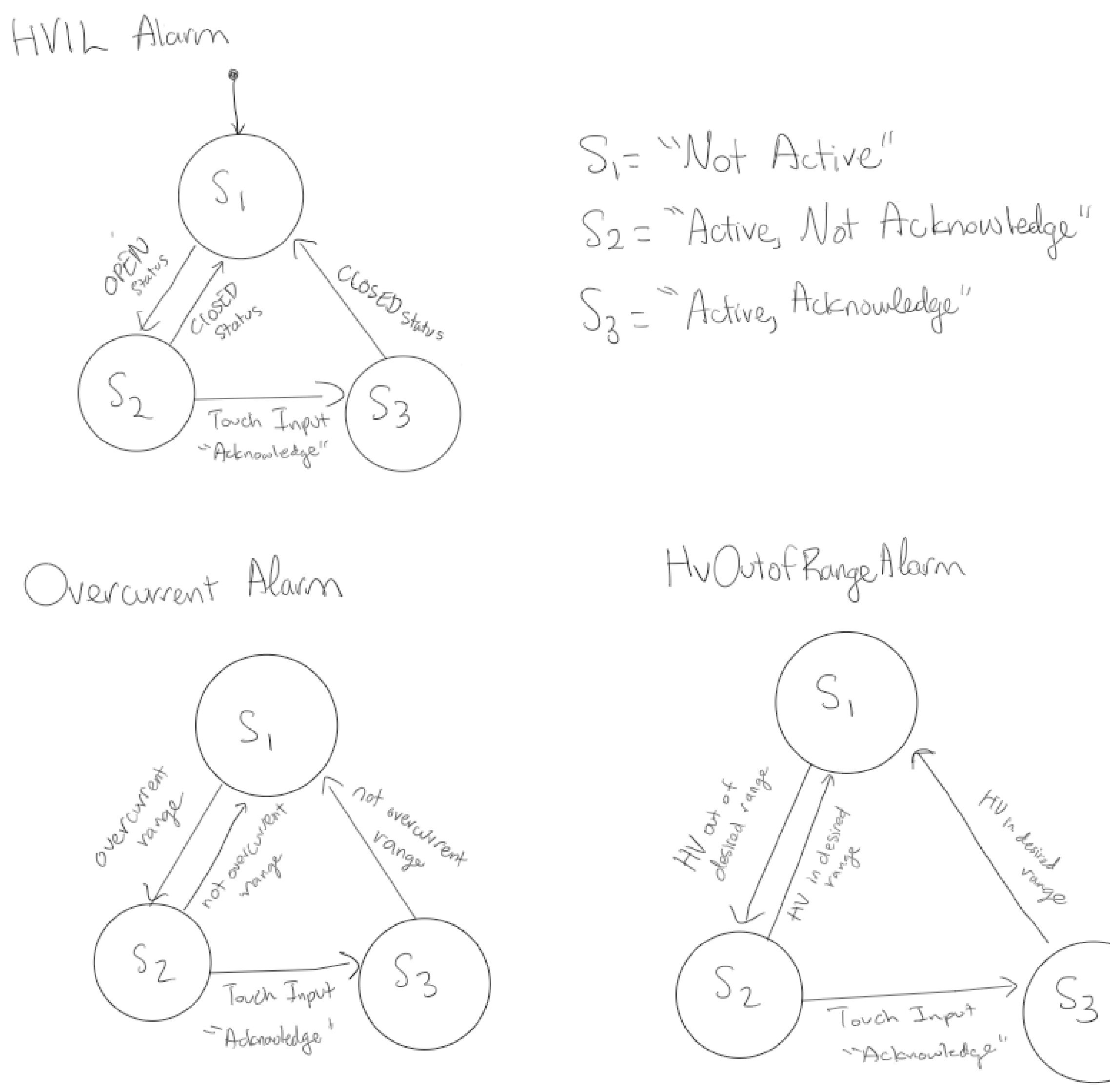


Figure 15 -- State Diagram for All Three Alarms

The contactor has only two states, open and close. Initially starts at open, and goes to close if the HVIL alarm is "Not Active" and the user input from the touchscreen is to turn the battery on. At close, it will transition back to open if the user input from the touchscreen is to turn off the battery or the HVIL alarm status is"Active, Not Acknowledge" or "Active, Acknowledge". This can be seen in Figure 16 below.

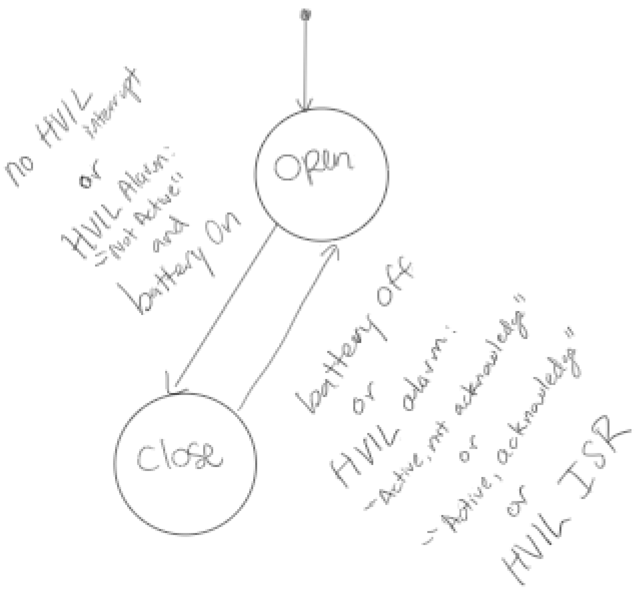


Figure 16 -- State diagram for contactor

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. However, the addition with this project is that if there is an active but non-acknowledge alarm then it will automatically transition to the alarm screen for all three states of the touch screen display task and not allow the user to transition away until the alarms are acknowledged. This can be seen in Figure 17 below.

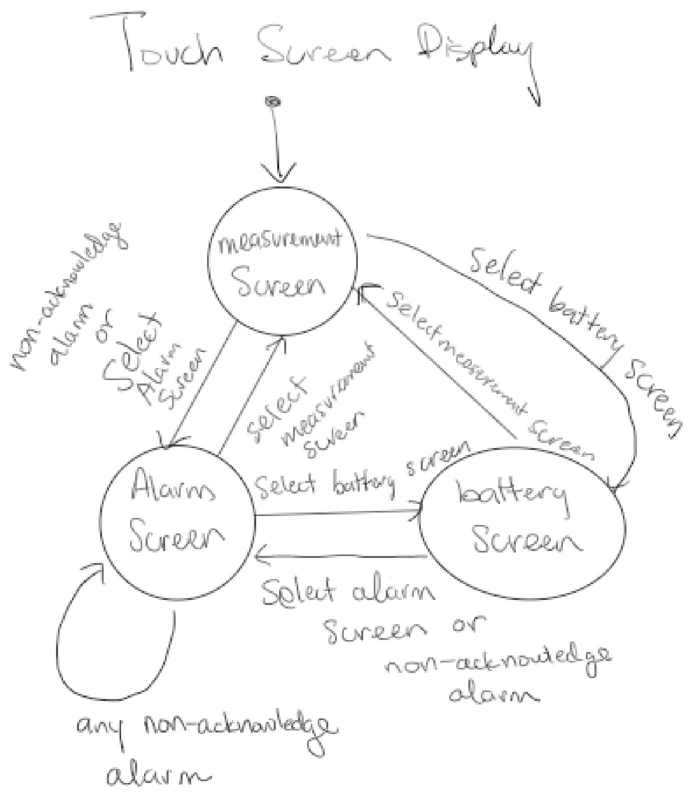

Touch Screen Display
measurement Screen
Select battery Screen
Select measurement Screen
non-acknowledge alarm or Select Alarm Screen
select measurement screen
Alarm Screen
Select battery screen
battery Screen
Select alarm Screen or non-acknowledge alarm
any non-acknowledge alarm

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 theTCB queue at a rate of 10 Hz

Measurement Function:
This function should update the temperature's state when the temperature potentiometer's value is changed, the HV current's state when the current potentiometer's value is changed, and the HV voltage's state when the voltage potentiometer's value is changed. It should also change the HVIL status through user interaction.

Alarm Function:
This function should update the HVIL alarm's state when the HVIL's status is changed, the overcurrent alarm's state when the HV current is in or out of the desired range, and the HVOOR alarm's state when the HV voltage is in or out of the desired range. All alarms states should change when acknowledged.

SOC Function:
This function should keep the SOC's state at 0.

Contactor Function:
This function should change the state of the contactorto open and turn off the contactor LED when the battery is turned off or when the HVIL's alarm is active and change the state of the contactor to closed and turn on the contactorLED when the battery is turned on and the HVIL's alarm is not active. 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 changingshould not be flickering. On the alarm screen, we should display a button to acknowledge alarms if and only if there are alarms to acknowledge.

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 allow the user to affect the contactor's state by turning the battery on and off using user input in the touchscreen. It should also allow the user to acknowledge alarms when needed and keeps the user from changing screens when there is an alarm to be acknowledged.

Timer1 ISR:
This function should toggle a flag at a rate of 10 Hz.

HVIL Interrupt ISR:
This function should change the HVIL alarm's status to active, not acknowledged and opens the contactor whenever the HVIL transitions from closed to open.

## 5.2 Test Coverage

Scheduler Function:
This function should run the tasks given in the TCB queue once per 0.1 seconds. We can test the frequency that the tasks run in to confirm they run at a 10 Hz rate

Measurement Function:
This function updates various measurements (temperature, HV Current, HV Voltage) through user input in varying ranges. We can test to see that the values are updating on change in the input per measurement and that the range is correct. 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 states based on changes in the values in the measurement task. We can test to see that the values are updating at the correct change of value per alarm.

SOC Function:
This function keeps the state of charge of the systemat 0. We can test to see that the value is always 0 and never changes.

Contactor Function:
This function changes the state of the contactor when the system flags it to do so and based on the state of the HVIL alarm's state. We can test to see that when this flag is to open the contactor or the HVIL alarm's state is active, the contactor opens and when this flag is to close the contactor and the HVIL alarm's state is not active, the contactor closes.

Display Function:
This function displays each screen and updates the state at the correct time for each aspect of the system. We can test that states are changing in real time when the physical inputs are changing, 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. We can also test that when we have one or more unacknowledged alarms, the acknowledge button is visible on the alarm screen, and not visible when there are no alarms to acknowledge.

Touch Input Function:
This function changes what screen is displayed, changes the contactor's state based on input from the user, and acknowledges alarms 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 and that the user cannot leave the alarm screen when there are unacknowledged alarms.

Timer1 ISR:
This function toggles a flag at a rate of 10 Hz. We can test that the timer flag is toggled at the correct frequency to make sure the hardware timer and its ISR are functioning correctly.

HVIL Interrupt ISR:
This function changes the HVIL alarm's status to active, not acknowledged and opens the contactor whenever the HVIL transitions from closed to open. We can test that the HVIL alarm's status does in fact transition to the correct state and that the contactor is opened when the function is called. We can also test that the function is only called when the HVIL transitions from closed to open.

## 5.3 Test Cases

Scheduler Function:
We tested using the millis() function and print outs to the Serial monitor how often each call to the scheduler function occurred. We found that the scheduler function runs every 100 ms plus or minus 1 ms, meaning that it runs at a 10 Hz rate.

Measurement Function:
We want to see that temperature updates whenever the input changes and that it ranges from -10ºC to 45ºC, so by changing the resistance of the potentiometer, we can see that the value of the temperature changes in real time and has the desired range. We want HV current to update whenever the input changes and that it ranges from -25A to 25A, so by changing the resistance of the potentiometer, we can see that the value of the current changes in real time and has the desired range. We want HV voltage to update whenever the input changes and that it ranges from 0V to 450V, so by changing the resistance of the potentiometer, we can see that the value of the voltage changes in real time and has the desired range. 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 is not active when the HVIL is closed, which we can observe. We also want the HVIL alarm to transition to active, not acknowledged when the HVIL is opened, which we also observe. We want the overcurrent alarm to be not active when the current is in the range (-5A, 20A), which we can observe. We also want the overcurrent alarm to transition to active, not acknowledged when the current is outside the range (-5A, 20A), which we also observe. We want the HVOOR alarm to be not active when the current is in the range (280V, 405V), which we can observe. We also want the HVOOR alarm to transition to active, not acknowledged when the current is outside the range (280V, 405V), which we also observe. For all the alarms, we want them to transition from active, not acknowledged to active, acknowledged when the acknowledgement button is pressed, which we can observe.

SOC Function:
We want to see that the SOC stays at a constant value of 0, which we can observe.

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 while the HVIL alarm is not active. When we touch the battery on button, we observe that the contactor's state only changes to closed and the contactor LED turns on when the HVIL alarm is not active, which is the desired behavior. We also want to change the contactor's status to open by turning the battery off through user input or when the HVIL alarm is active. When we touch the battery off button, we observe that the contactor's state changes to open and the contactor LED turns off, as is the case when the HVIL alarm transitions to be active, 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 the values don't blink when they aren't being updated, showing that we are observing the intended behavior. When we are on the alarm screen and there are no alarms to be acknowledged, we can see that the acknowledgement button is not visible as intended. And when one or more of the alarms is notacknowledged, the buttons pops up on the display as intended.

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 displaythen shows the correct page. We tested the measure, alarm, and battery buttons and they all transitioned the display to their respective screens. When we have an unacknowledged alarm, the function automatically takes us to the alarm screen and doesn't allow the user to transition without acknowledging the alarm(s), as intended. 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, given the HVIL alarm is in the correct state for each transition. Finally, we tested that the acknowledge button on the alarm screen does the correct thing. When we have one or more alarms that are active, not acknowledged on the screen and we press the acknowledge button, the not acknowledged alarms transition to active, acknowledged, which is the correct behavior.

Timer1 ISR:
We read the timer flag and untoggle it in the scheduler, so whenever the scheduler is run, that means the flag was toggled recently. The scheduler runs at a rate of 10 Hz, meaning the Timer1 ISR function is called at the same rate and toggles the flag every time, meaning it has the correct behavior.

HVIL Interrupt ISR:
We want this function to be called only when the HVIL transitions from closed to open. We observe that the function is in fact called when the HVIL transitions from closed to open, but also sometimes in the other direction. This error is explained and discussed more in-depth in section 6.2 of the lab report. We also want this function to change the HVIL alarm's state to active, not acknowledged and open the contactor when called, which we can observe when we toggle the HVIL switch.

## 6.0 Presentation, Discussion, and Analysis of the Results

All modules and tasks were completed as they passed all the test cases required, except for the HVIL interrupt task. The tasks that worked executed without any errors and resulted in flawless data flow and intertask communication. The display functionality updated the values on change for each of the values measured in this project. The measurement screen showed the correct values as they updated in real time along with the alarm screen updating the alarm state for each of the alarms in the correct conditions. Additionally, the connection between the touch screen display, the HVIL, and the contactor worked seamlessly as inputs on the touch screen and the state of the HVIL combined to result in state changes on the battery. Our issues were with the HVIL interrupt task triggering at inappropriate times, leading the HVIL alarm's status to switch to an undesirable state for less than a second. This caused the screen to automatically transition to the alarm screen on some of the switches of the HVIL state from open to closed. Overall, we were mostly successful in developing our battery management system further. We managed to implement a dynamic task queue, use hardware timing, introduce interrupts to our system, albeit with bugs, and incorporate analog reading of physical values in our system.

## 6.1 Analysis of Any Resolved Errors

When updating the HVIL alarm's state in the HVIL ISR, there we encountered some interesting behavior. The display printed corrupted data as the HVIL alarm's status, and after some debugging, we discovered that the HVIL alarm status variable was being set incorrectly in the HVIL ISR. Our guess is that this is caused by some intricacies with the status being a volatile char* variable. This caused us to change the alarm status variables to be ints, as they should be easier to work with when volatile. After making this adjustment and switching of all uses of the alarm statuses towork with them as itns, the system ran as intended, albeit with the below issues.

## 6.2 Analysis of Any Unresolved Errors

Our group had issues with the HVIL interrupt running at inappropriate times. When we opened the HVIL, it would run twice and when we closed the HVIL, it would run, changing the HVIL alarm's status to "active, not acknowledged" before transitioning to "not active". Since we want the system to switch to the alarm screen whenever we have a non-acknowledged alarm, the screen switches to the alarm screen when we close the HVIL sometimes as well. This error only occurred on one team member's system, ruling out any software issues. Attaching an oscilloscope found no obvious issues, as the rise and fall of the signal looks smooth and lacks bouncing. Our group could try using a different switch or perform more in-depth tests to find the conditions where there is no unwanted interrupt when closing the HVIL, as the intended behavior does occur sometimes.

## 7.0 QUESTIONS

- What is the resolution of each of the [0, 5V] analoginputs being simulated via potentiometer (HV voltage, HV current, and Temperature)?

  Since the analog input returns a 10-bit value, that means it would range from 0-1023.
  The temperature ranges from -10ºC to 45ºC, so the range is 55ºC and the resolution is 0.0538ºC / analog value.
  The current ranges from -25A to 25A, so the range is 50A and the resolution is 0.0489A / analog value.
  The voltage ranges from 0V to 450V, so the range is 450V and the resolution is 0.440V / analog value.

- List of all inputs and outputs categorized as either digital or analog signals
  - Inputs:
    - Temperature value (Analog)
    - Current value (Analog)
    - Voltage value (Analog)
    - HVIL status (Digital)
  - Outputs:
    - Contactor status (Digital)

## 8.0 Conclusion

As the purpose of this project is to modify a battery management system that was implemented in the last project with interrupts, dynamic task queue, different functionalities, state changes, and much more, there was a tremendous amount of moving parts to this project. The project structure started with redevelopment of all UML diagrams in order to understand the structure and intertask communication between the different tasks and modules. We examined the differences between the projects and what needed to be added or modified in each of the functions in order to produce the desired system. Programming the different data flows between the ATMega pins, HVIL, contactor, HV Current, HV Voltage, and Temperature was challenging. However, the project was able to teach and demonstrate a huge amount of understanding of the material based on interrupts and understanding how it is implemented in code and how it operates. Besides interrupts and ISRs, this lab was able to teach the implementation of a dynamic task queue along with alarm flags for state transitions. We don't have many recommendations or suggestions for improving this project as the topics and concepts that are covered are taught in class and are building blocks for more complicated projects. Overall, this project provides a great learning experience into programming interrupts, hardware timing functions, dynamic task queue, along with analog inputs from other devices.

## 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:**

```
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
initialize the hardware timer to run at 10 Hz
set up the HVIL interrupt routine
```

**Scheduler Task Pseudocode:**

```
if the hardware timer flag tells us it’s time to perform tasks:
    set the current TCB to the head of the linked list
    while the current TCB isn’t null:
        perform the current TCB’s task
        move on to the next TCB
```

**Measurement Task Pseudocode:**

```
updates HVIL state by reading from the HVIL input pin
update the temperature’s value by reading from the temperature pin
update the HV current’s value  by reading from the current pin
update the HV voltage’s value  by reading from the voltage pin
```

**Alarm Task Pseudocode:**

```
update the HVIL alarm’s state based on the HVIL’s status
update the overcurrent alarm’s state based on the current value
update the HVOOR alarm’s state based on the voltage value
```

**SOC Task Pseudocode:**

```
set the SOC value to 0
```

## Contactor Task Pseudocode:

```
if we want to close the contactor and the hvil alarm is not active:
    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 or the hvil alarm is active:
    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
    if the temperature changed or we just switched to this screen:
        display the updated temperature state
    if the current changed or we just switched to this screen:
        display the updated HV current state
    if the voltage changed 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
    if there are alarms to acknowledge and the button is not drawn:
        draw the acknowledgement button
    if HVIL alarm status changed or we just switched to this screen:
        display the updated HVIL alarm state
    if the overcurrent alarm status changed or we just switched to this screen:
        display the updated overcurrent alarm state
    if HVOOR alarm status changed 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'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
    if we're on the alarm page:
        if we pressed the acknowledge button:
            acknowledge all unacknowledged alarms
    if there are any alarms to acknowledge:
        change the page to the alarm page
    else:
        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
```

**Timer ISR Pseudocode:**

```
set the timer flag to true to signify that it's time to run tasks
```

**HVIL ISR Pseudocode:**

```
set the hvil alarm status to active not acknowledged
open the contactor
```