**Lab 3: Climate Control and Room Occupancy Final Design**

Eric Pires

John Lamanuzzi

Richard Harrison

ECE 388

Team 3

Lab Date: November 23, 2016

Due Date: December 15, 2016



**Table of Contents**

[Introduction 2](#_Toc469507293)

[Methods and Testing Procedures 3](#_Toc469507294)

[Board Population 3](#_Toc469507295)

[Component Evaluation 5](#_Toc469507296)

[System Evaluation and Model Design 6](#_Toc469507297)

[Results 8](#_Toc469507298)

[Discussion 16](#_Toc469507299)

[Conclusion 17](#_Toc469507300)

[Reflection 19](#_Toc469507301)

[Post Mortem 20](#_Toc469507302)

[References 21](#_Toc469507303)

[Appendix 22](#_Toc469507304)

Abstract

The goal of this laboratory was to populate and evaluate the final board design of the climate control and room occupancy system to then be demonstrated for the final presentation. Components were soldered and tested to ensure their functionality within the system. The User Acceptance Test (UAT) was followed to ensure the system was fully operational. The system was then implemented in a scaled environment and presented in class.

# Introduction

The objective of this lab was to finish building the final system and evaluating its functionality in comparison to the UAT. The following tools were used in this design phase:

* Hakko FX-888D Soldering Station
* AM Scope
* Etekcity MSR-R500 Digital Multimeter
* Sparkfun USBtiny AVR Programmer

The system under test utilized the following components that were later tested:

* 16x2 LCD screen
* DHT-22 Temperature and Humidity Sensor
* 2x Tactile Switch SPST-NO
* 2x Infrared Obstacle Avoidance Sensor
* 2x Uxcell 12VDC 0.25A CPU fan
* Honeywell HCE100R Heat Bud Ceramic Heater
* 40W Desk Lamp

Following the population of the PCB, the USBtiny AVR programmer was configured to upload the previously written firmware from the prototyping phase to ensure each component’s functionality. After confirming each major component’s functionality, the final firmware was uploaded to the board and evaluated in accordance to the test plan. The system was then built inside a built model to emulate a closed environment.

# Methods and Testing Procedures

## Board Population

Following the arrival of the boards and purchased components, the population phase began. To start, the ATmega328p microcontroller was put under the AM scope to be soldered. This component contained 32 total pins (8 on each side). To solder this device, the corner pins were first soldered to keep the device in place. Once this was done, each side was soldered by using the chisel-tipped iron to run solder on all pins. To separate the bridged pins, soldering wick was heated and placed over the pins. Once removed, the solder in between the pins was stripped away. This process was repeated for the remaining three sides. Once all pins were soldered, test procedures 2-3 found in Table 1 were followed to ensure the device was properly applied.

The next component was the voltage regulator. This device contained 3 pins to solder. The ground plane required more solder to flow underneath the package. After these components were soldered, the 0805 packages were soldered. This included resistors, capacitors, and LED’s. Note that the LED’s required an additional testing procedure due to their polarity. This can be found in Table 1. The next step was to finish soldering the remaining surface mount components. This included the crystal, diodes, transistors, and push buttons.

The last step in populating the board was the through-hole components. These included the DC jack, relay sockets, and headers. This step was the simplest since it required less precision to solder compared to the surface mount devices. Once more, the procedures listed in Table 1 were followed to ensure these components were properly soldered.

*Table 1*

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test Name | Test Description | Procedure Description |
| 1 | Polarity Check | Is the device placed in the correct orientation specified on the silkscreen? (Done for diodes, LEDs, header pins,etc.) | Read the datasheet of the given device to find out how the device polarity is marked. For the LEDs, use the multimeter’s polarity mode to find the polarity by applying the probes on both sides of the device. If the probes are correct, the LED should illuminate. |
| 2 | Solder Check | Does the applied solder make contact with the pin and pad? | Through eye inspection (or through the AM scope) inspect the soldered joint to ensure that there is enough solder applied to it. Remove any excess solder using the chisel-tipped iron. |
| 3 | Continuity Test | Is the pin properly soldered to the pad and does it make a connection to its traced path? | Use the multimeter’s continuity mode to supply a low current through the connected probes. Attach one probe to the soldered pin and another onto a pad or component in series with the pin. The multimeter should beep if the path is connected properly. |

Figure 1 of the Appendix illustrates the board from its initial look to the final look with all components soldered. The final test was plugging the power connector to the board. The power led was observed once the board was supplied power.

## Component Evaluation

After populating the board, the programming phase began. Since the board lacked a built-in debugging solution, the ISP pins were connected to the USBtiny programmer. To communicate with the device, WinAVR drivers were installed to detect the connected programmer. Using Arduino’s “Burn to bootloader” option, the “blink” Arduino sketch was uploaded, adjusting the led pin to be the MAX led. After confirming the programmer’s functionality, the prototype code was uploaded to the microcontroller to ensure each of the major components worked as designed. The following test procedures were done for each major component:

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test Name | Test Description | Procedure Description |
| 1 | Initialization Test | Will the AVR programmer upload code to the MCU? | Upload the modified “Blink” sketch from Arduino onto the board. Observe if the MAX led turns on and its timing. |
| 2 | LCD Test | Will the LCD display a written message? Does this message get cleared and re-written upon a system reset? | Upload the Arduino “HelloWorld” sketch onto the MCU. Hit the reset button to see if the message re-initializes. |
| 3 | DHT Test | Will the DHT read the current temperature in the room and print on the LCD? | Upload the DHT Arduino test code to the MCU. Compare the measure temperature to the temperature in the testing environment. |
| 4 | Button Test | Will the button respond accordingly to the user’s input? | Upload the button test Arduino sketch to the MCU. Press each button and see if the count responds accordingly. |
| 5 | IR Test | Will the IR sensor increase/decrease the room count upon triggering each sensor? | Upload the room occupancy Arduino code to the MCU. Trigger each sensor to change the room count accordingly. Print out the room count onto the LCD. |
| 6 | Fan Test | Will the fan header power the 12V fan accordingly? | Enable the fan pin high on an Arduino sketch. Plug in the 12V fan to the labs power supply and measure the current draw though the fan. |
| 7 | Relay Test | Will the light relay (RLY\_1) and heater relay (RLY\_2) activate when sent a signal from the MCU? | Send a high signal to the relay ports and observe whether or not the relay switches by observing both the relay click and led indicator. |

Since the microcontroller defaults to using the internal clock source, the fuse bits were changed to reconfigure the device to use the external clock of the board and to enable serial programming. The following arguments were passed for the AVRdude command:

**avrdude -U lfuse:w:0xff:m -U hfuse:w:0xdf:m -U efuse:w:0xff:m**

Since the final code was written in Atmel Studio, a new method of programming was required. To do this, AVRdude was invoked in the command prompt to upload the given hex file to the board. This method was done as explain in Sparkfun’s AVR programing guide. The following command uploaded the code to the device:

**avrdude -c usbtiny -p atmega328p -U flash:w:Climate\_Control.hex**

## System Evaluation and Model Design

Having tested all the individual components, the final step was to go through the UAT. With the final code uploaded and ready to evaluate, the following UAT procedures were done:

|  |  |
| --- | --- |
| **6. UAT Requirements-Based Test Cases** | |
| **ID** | **Test Cases** |
| 6.1 | **Procedure:** Measure Voltage Regulator  **Expected Results**: Voltage and current input is 12V, 1A while outputs are 5V, 200mA |
| 6.2 | **Procedure:** Measure Desk Lamp control  **Expected Results**: Desk lamp is provided with a maximum of 363mA when the relay is activated. |
| 6.3 | **Procedure:** Measure Heater control  **Expected Results:** Heater is provided with a maximum of 1.55A on the low setting, 2.27A on the high setting. |
| 6.4 | **Procedure:** Measure Fan control  **Expected Results:** MOSFET triggers the fan load with logic signal. (High for on, Low for off). Fans are provided with at most 250mA. |
| 6.5 | **Procedure:** Increase Set Temperature  **Expected Results:** The UI will instantly update the set temperature value accordingly to the number of presses from the user. The heating relay will activate and power on the personal heater (Defaulted to the low setting). The relay will deactivate once the set temperature is matched with the currently read temperature. |
| 6.6 | **Procedure:** Decrease Set Temperature  **Expected Results:** The UI will instantly update the set temperature value accordingly to the number of presses from the user. The cooling fan will be activated and will terminate when the currently measured temperature matches the set value. |
| 6.7a | **Procedure:** Add new entity into empty environment  **Expected Results:** IR entrance sensor will trigger. Light relay will activate and power the desk lamp. |
| 6.7b | **Procedure:** Add excess of entities into environment  **Expected Results:** MAX LED is enabled |
| 6.7c | **Procedure:** Decrease entities under maximum allowance  **Expected Results:** IR exit sensor will trigger. MAX LED is disabled |
| 6.7d | **Procedure:** Remove all active entities in environment  **Expected Results:** Light relay is disabled. Desk Lamp will turn off |

Once this was confirmed, the system model design began. The system would be incorporated inside a 72-qt tote. Incisions for the LCD, fans, entrance, and power cables were made to the tote. The LCD and fans were then mounted and drilled inside the tote. The IR sensors were placed on a small wooden block close to the entrance. They were then drilled down to the block, which was then drilled down to the tote itself. The board was placed on an elevated platform and the required cable routing was done for each external component. This cabling included the LCD, fans, heater, and IR sensors. The UAT was done once more to ensure the system would operate inside the environment. The final design can be found in Figure 2.

# Results

Upon the completion of all soldering, the procedures of Table 1 were performed and were mostly successful. Cold solder joints were found on the down button and voltage regulator. This was quickly resolved by applying more solder onto the component. When powering on the system, the power led illuminated and all components were connected and functional.

The component evaluation was performed for each major component. The only issues found involved the LCD and the crystal. The LCD was not displaying any values. Due to prior issues seen in previous prototypes, the first hypothesis for this was that the Vo pin contained a high resistivity. To adjust this, a new resistor replaced the 1K resistor found at the pin. The quick adjustment fixed the issue altogether. The other issue found was the crystal. At first, the blink program contained an incorrect timing during the led blinking. This issue was resolved by adjusting the fuse bits to accept the on-board crystal. The table of results can be found below.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Test Name | Test Description | Result |
| 1 | Initialization Test | Will the AVR programmer upload code to the MCU? | Attempt 1: Fail  Code uploaded by timing was off by a factor of 30.  Attempt 2: Pass (Fuse bits adjusted to accept 16MHz crystal for correct timing) |
| 2 | LCD Test | Will the LCD display a written message? Does this message get cleared and re-written upon a system reset? | Attempt 1: Fail  No visible output on LCD  Attempt 2: Pass  150Ω resistor replaced 1k resistor on Vo pin. LCD properly displays “Hello world” |
| 3 | DHT Test | Will the DHT read the current temperature in the room and print on the LCD? | Pass |
| 4 | Button Test | Will the button respond accordingly to the user’s input? | Attempt 1: Fail  Incorrect usage of interrupt used on PCINT pin  Attempt 2: Pass  Code fixed to use pin change interrupt on respective pin |
| 5 | IR Test | Will the IR sensor increase/decrease the room count upon triggering each sensor? | Pass |
| 6 | Fan Test | Will the fan header power the 12V fan accordingly? | Pass |
| 7 | Relay Test | Will the light relay (RLY\_1) and heater relay (RLY\_2) activate when sent a signal from the MCU? | Pass |

The ten tests conducted during the demo were the same ten outlined in the UAT: 6.1 through 6.6 and 6.7a through 6.7d. They consisted of the ten main functions of the climate control and room occupancy system as a whole, with corresponding electrical measurements where applicable. All of the tests (save for 6.1), required direct user input; the tests were conducted and interpreted according to the UAT Scope, so some leeway was given when necessary, and some tests were not conducted. The UAT Scope states that the temperature of the system can be controlled with the UI (that being the up and down buttons on the system board), the room occupancy count is a rough estimate, and a light will shine indicating an occupancy threshold. Out of scope measurements and modifications for the project were the adjustment of temperature outside the given range (that being 10°C and 30°C), changing the humidity of the closed environment and ensuring perfect room occupancy count (a much more robust and expensive system would be required for perfect accuracy).

Test 6.1

“Voltage and current input is 12V, 1A while outputs are 5V, 200mA.”

This test was conducted with a multimeter, and the result of the test was that the input from the DC jack was 12V, 1A (as per the DC jack specifications), and the output from the VRM was 5V, 200mA. An incorrect value here could cause the microcontroller or other devices to malfunction or sustain damage, and cannot be changed by the user, as per the UAT Scope.

Test 6.2:

“Desk lamp is provided with a maximum of 363mA when the relay is activated.”

This was the first test that required user input. When any occupancy is reached, the desk lamp would turn on. This would mean that two wall sockets would be required for the demonstration, but thankfully, even when the relay isn’t switching to a device, it will still give an audible click. This meant that the desk lamp did not have to be connected to the system. When the occupancy count reached one, the relay for the lamp clicked, indicating the lamp would shine if it were connected. Because the lamp was not connected during the demonstration, current was not measured. However, further testing indicates that the current supplied to the lamp was 363mA. This test is also related to tests 6.7a-6.7d.

Test 6.3:

“Heater is provided with a maximum of 1.55A on the low setting, 2.27A on the high setting.”

This required the user to increase the set temperature to a setting above the current temperature. At this point, the heater, already set to either high or low, would run until the desired temperature is met. This was controlled by the relay again, so it was feasible to have the heater disconnected from the system and listen for the click, but the heater was important for other, more general tests. Because of this, the heater was indeed used and measured during the demo. The current supplied on low was 1.55A, and when switched to high, 2.27A was supplied. This test is also related to test 6.5.

Test 6.4:

“MOSFET triggers the fan load with logic signal. (High for on, Low for off). Fans are provided with at most 250mA.”

This required the user to decrease the set temperature to a setting below the current temperature. As described by the expected results of the UAT Test Cases, the two fans were controlled by a MOSFET instead of a relay. The specification for one fan is that it would draw 250mA, but as two were used, the current supplied to the two fans should be 500mA. When the fans were activated, the current draw of the two combined was 420mA, measured with a multimeter. This test is also related to test 6.6.

Test 6.5:

“The UI will instantly update the set temperature value accordingly to the number of presses from the user. The heating relay will activate and power on the personal heater (Defaulted to the low setting). The relay will deactivate once the set temperature is matched with the currently read temperature.”

This test focused on the user directly interfacing the system, and ensuring that the temperature is increased by the correct amount. It required correct interfacing between the button, microcontroller, and display, and kept in mind the set temperature’s given upper bound of 30°C. Given a button press, the set temperature increased by one, and subsequent presses would do the same, indicating proper incrementation. This was backed up with the system’s interfacing with the heater, which indeed turned on when the set temperature exceeded the room temperature, and turned off when the set temperature met or fell below the room temperature.

Test 6.6:

“The UI will instantly update the set temperature value accordingly to the number of presses from the user. The cooling fan will be activated and will terminate when the currently measured temperature matches the set value.”

This test was very similar to test 6.5, in that it ensured proper interfacing with the UI, but this time the down button was tested, and in a similar way. It once again kept in the set temperatures given bounds, this time the lower bound of 10°C was considered. On a button press, the set temperature decreased by one, and subsequent presses would continue to decrement the setting according to the amount of presses. This was backed up by the system’s interfacing with the two fans, which indeed turned on when the set temperature fell below the room temperature, and turned off when the set temperature met or exceeded the set temperature.

Test 6.7a:

“IR sensors will trigger, indicating entrance. Light relay will activate and power the desk lamp.”

This test focused on the proper functioning of the IR sensors, and the relationship between the sensors, the light, and the display. For the purposes of the demonstration, the humidity reading was replaced with the room count. The reason for this is because the four measurements that are possible (current temperature, set temperature, humidity, and room count) cannot all be displayed on-screen at once; there is not enough space on the display for all four values. Because of the size of the test environment, hands were waved in the desired direction to simulate an entity entering or exiting; toward the center of the room indicated entrance, away from the center indicated exit. In this case, an entrance was simulated, and the sensors detected this, incremented the count, and activated the relay (if the lamp was connected, it would turn on at this point).

Test 6.7b:

“IR sensors will trigger. MAX LED is enabled.”

When an excess of entities enters the environment, the LED on the project board labelled “MAX LED” would shine, indicating this excess. For the demonstration, the maximum of entities was three, as it would be an easy value to reach. Of course, this value could be easily reprogrammed. As this test follows 6.7a, the procedure was similar, except instead of simulating just one entity, the goal was to simulate three or more. After three entities were simulated, counted by the display and by the tester, MAX LED indeed lit up, indicating such excess.

Test 6.7c:

“IR sensors will trigger. MAX LED is disabled.”

When the count of entities in the environment is reduced below the threshold, the previously lit MAX LED would subsequently shut off. This test was conducted in the same way as Test 6.7b, except it simulated entities leaving the environment, rather than entering, and implies the success of 6.7b. After 6.7 was tested and confirmed to be functional, the exit of the entities were simulated, decrementing the count accordingly. When the count fell below the threshold, MAX LED turned off.

Test 6.7d:

“Light relay is disabled. Desk Lamp will turn off.”

This test is similar to that of 6.7a, except instead of simulating the first entity entering the environment, this test simulated the last entity leaving the environment. In this test, the final exit should trigger the lamp to shut off. The lamp was not connected to the system, so the proper deactivation of the relay was considered instead. The exits of the entities were simulated such that there would be zero in the environment, and the relay shut off, indicating that the lamp would also had it been connected to the system.

With the results of each test case considered, there were still some minor faults regarding an error-free operation of the product. One issue that occurred was the proper updating of the display to reflect the measurements and settings to be shown. Occasionally, the LCD would display random characters (sometimes mixed in with the measurements) when any of the three values were updated, requiring a system reset. This was also possible on reset, requiring subsequent system resets until the LCD displayed correctly. While the board had a reset button, mitigating this issue somewhat, it was determined to be a slight error in the code that interfaced the microcontroller with the display.

The second issue was regarding the IR array, a decision changed from exit and entrance doors very close to the demonstration day. Because of the way the array is set up, the order in which they are tripped will determine if the entity is entering or exiting the environment. However, if only one sensor of the two is tripped, unintended functions occurred. If the first (i.e. closest to the door) sensor was tripped first, the count will increase regardless of whether or not the second (ie. closest to the center) sensor is tripped. This is partly considered in the main control loop, but only such that tripping both of the sensors (i.e. fully entering or fully exiting) will change the count according to the order.

The third issue was regarding the accuracy of the temperature readings from the DHT22. Although it is more precise than the DHT11, there is still some degree of inaccuracy that must be considered. Although the humidity of the system was not tested (again, this test fell out of favor because displaying the occupancy count would be more helpful for the demo), this too would also have some inaccuracy. The temperature reading is good for -40-80°C, with ±0.5°C of accuracy. This means that if another sensor measured the same place as the DHT22, it could vary by a full 1° at the very worst case. During the testing of the heater and fans, another temperature sensor, provided by the instructor, was used to ensure that the heater and fans were functioning properly, and it uncovered that the system was reporting values that contradicted the secondary sensor by more than the degree of accuracy; in some cases, the two measurements varied by up to 5°, a much larger discrepancy than advertised. Granted, the secondary sensor used also has its own degree of accuracy, which was not provided by the instructor. However, it is unlikely it has ±4.5°C of potential inaccuracy, meaning the worst cases measured should still have been closer. All test cases can be found in the table below.

|  |  |  |
| --- | --- | --- |
| **6. UAT Requirements-Based Test Cases** | |  |
| **ID** | **Test Cases** | **Results** |
| 6.1 | **Procedure:** Measure Voltage Regulator | Input: 12V, 1A. Output: 4.98V, 194mA |
| 6.2 | **Procedure:** Measure Desk Lamp control | Current Draw: 363 mA |
| 6.3 | **Procedure:** Measure Heater control | Current Draw: 1.475 A |
| 6.4 | **Procedure:** Measure Fan control | Current Draw: 210mA |
| 6.5 | **Procedure:** Increase Set Temperature | UP(->) button increased set temperature by 1 degree for every press. System did not exceed specified maximum value (30) |
| 6.6 | **Procedure:** Decrease Set Temperature | DOWN(<-) button decreased set temperature by 1 degree for every press. System did not exceed specified minimum value (20) |
| 6.7a | **Procedure:** Add new entity into empty environment | Light Relay activated, count increased to 1 |
| 6.7b | **Procedure:** Add excess of entities into environment | Max led activated, correct number of IR triggers shown on LCD |
| 6.7c | **Procedure:** Decrease entities under maximum allowance | Max led deactivated, correct number of IR triggers displayed |
| 6.7d | **Procedure:** Remove all active entities in environment | Light Relay deactivated. Count is at 0 and will not decrease to negative value. |

# Discussion

The goal of this project was to develop an embedded system capable of measuring and displaying temperature within a given environment. The system was also responsible for keeping a count of the room occupancy in order to send an alert whenever a maximum was reached. The design demonstrated in this report used a DHT22 temperature and humidity sensor to communicate with a 16x2 LCD screen to display the currently measured temperature. The user interacts with this system through two tactile switches that change the status of the system. Whenever the user selects a higher temperature than currently measured, the system will enable a connected heater system through a relay. When the user selects a lower temperature, the system will enable a cooling fan. This process was interrupt driven to ensure the user’s input took precedence. The user also interacts with the system through the IR sensors. These sensors, paired side by side. Would act as a trip wire whenever an entity enters the environment. When activated, this device increases or decreases the current count of entities in the room. When a maximum is reached, the system will enable an led which signifies an alert system. The system also reacts when no entities are present. It disables the relay that controls the light source of the environment. The system flowchart found in Figure 3 of the Appendix demonstrates this functionality.

When populating and evaluating the components of the system, one issue was noticed with the selected components. The LCD contrast pin (V0) was tied to a 1K resistor to ground. The resistance found at this pin was too low for the LCD to display characters. This was detected when measuring the voltage drop caused by this resistor (R3). To resolve this, the proper resistance value used in the prototyping phase (150Ω) was acquired and replaced the 1K resistor. This quick change solved the issue and allowed the climate system to properly communicate to the LCD.

The biggest issue faced in the final implementation was setting the fuse bits. Initially, the fuse buts were set to accept an external clock source at one pin. This mishap caused the MCU to be unprogrammable since the device was not registering a clocked pulse signal. Through Professor Viall’s guidance, the clocked pulse was provided to the MCU through the function generator in order to reset the fuse bits to the proper clock settings. The datasheet for the MCU was carefully read to select a proper fuse bit to work with the 16MHz crystal.

Programming to the board took more work than initially perceived. Since the climate system only contains a 6-pin programming header, the USBtiny was required as an external programmer. Using the device required some research and new system drivers to connect to a computer. The guide offered by Sparkfun helped make this process brief and easy to follow.

# Conclusion

Everything that was done this semester culminated in the demonstrations of the final product for the climate control and room occupancy system. It was held against the ten UAT Test Cases for the general functionality of the product as a whole, and although there were a couple of small issues that were encountered during the demo, the session as a whole went quite well. The first six tests that were conducted had to do with the general function of the product, ensuring the product worked in general. These six tests entailed the functionality of all the sensors and actuators of the project, as well as the input and output voltages of the voltage regulator.

The sensors and actuators tested in this section were the two buttons, the DHT22, the PIR array, the desk lamp, the heater, and the two cooling fans. The testing of the sensors and actuators meant that related processing had to be tested, as well. The related processing saw focus during the testing of the buttons (and therefore, the DHT22) and the PIR sensors, the latter of which got their own battery of tests. The tests for the PIR array were the correct function of all possible scenarios: entities entering and exiting, as well as interaction with the occupancy threshold; if the threshold is met, the MAX LED will shine, indicating this threshold. When entities exit the environment such that the amount falls below the threshold, the LED will turn off, and when there all entities exit, the lamp will shut off.

However, three minor problems were encountered during the demo session. The first problem was the LCD would occasionally print gibberish, which could sometimes require multiple resets. The second problem was that the PIR array did not take in account entities that only tripped one sensor, which would have the unintended outcome of incrementing or decrementing the count depending on which one was tripped. The final problem was that the DHT22’s temperature reading did not always match the thermal probe that was used for testing the temperature readings. Although these issues are documented in the Postmortem in more detail, along with other issues the group faced, all of these issues could be remedied fairly easily, with extra delay in dealing with printing to the LCD and resetting the system when required, added code that takes in account of only one PIR sensor tripping, and the use of a second temperature sensor for better accuracy, respectively. All in all, this design project has been a very interesting and informative look at the world of designing a real product from scratch, and the development of said product from start to finish in the accelerated timeframe of only a couple of months.

# Reflection

Over the course of the semester, this embedded design project has prepared us for our future careers. We have gone through the life cycle of an embedded systems design including prototyping, board design and evaluation. Skills such as soldering, EAGLE design and test analysis were learned. Other skills previously learned through the course of our studies, such as circuit theory and digital logic were tested and improved on. In the prototyping phase, we learned how to choose the correct components for our project through research and repetitive testing. Fundamental skills like breadboarding and soldering were improved on throughout the design process. Board design thorough EAGLE was rigorous but incredibly informative. The system firmware was another key challenge. During the prototype stage, Arduino libraries were used, however these were replaced with C libraries supported on the Atmel Studio environment. The most valuable knowledge gained throughout this project was test driven design. By documenting clear test matrices and consistently working through them, the overall design process felt much simpler to follow and debug for any issues. The work done to document these tests was extensive and time consuming, but felt extremely helpful to achieving the results of this design project.

One board improvement that could have been done was the placement of the push buttons. Accessing the buttons was tedious and impractical for a real design application. However, if the buttons were placed on the back of the board, then the board could have been enclosed face down with the buttons sticking out. This could have made accessing the buttons much easier. The overall size of the board was not an issue. We feel in a real climate control application; the board would fit nicely inside a plastic enclose containing the LCD and user buttons. In fact, we observed a commercial climate control system board and its size was much larger than our design. The board only had about 10 input pins that were routed through the wall.

We feel that many firmware improvements could have been made. One case point being the room occupancy firmware. The implemented firmware was very simple and did not account for false readings. The firmware could have easily been modified to ensure that no change was made if only one sensor was triggered. The temperature reading could have been improved on as well. The firmware invokes the truncate function to compare integer temperature readings rather than floating decimal values. Doing so reduced the accuracy of the reading dramatically and in turn caused more on board memory to be used since an additional header file was needed. The LCD could be repurposed to display more decimal values for the temperature readings.

Looking at the size of the final uploaded firmware, we believe that the ATmega328p might have been an excessive component that could have been replaced for cost management. We feel the cheaper ATmega8 could have achieved a similar function with only about 5KB of used storage. Overall, the project drastically changed the way we approach system design and has definitely provided great insight on future expectations as we approach the final years of our engineering education.

# Post Mortem

The final system completed the task of providing the user with a climate control and room occupancy system. The DHT22 temperature sensor worked as expected, but the utilized firmware truncated the readings and resulted in slightly inaccurate readings being displayed. The biggest issues came from the occupancy control. During the testing phase, the IR sensors were triggered so that if an entity entered the doorway and left, the room count would increase by one, despite having no actual entities inside. The written firmware could have been improved to avoid this case. The LCD worked as expected except in some few cases when the system was reset. Due to the refresh rate of the LCD compared to the reset time, the board reset too quickly and the LCD did not have time to completely clear the screen. A solution to this could have been to implement a longer delay on reset to allow the LCD time to fully clear. As previously mentioned, the control buttons could have been placed on the back of the board for convenience of access. The issues found in our final design were generally minor and could have been easily fixed with a firmware update. The occupancy count was never expected to be fully accurate, but its implantation was much more practical than the previously designed solution. This involved creating two passages, one to enter only and the other to exit only. Perhaps with a more expensive occupancy solution, the system might have done a better job in detecting room occupancy. Such a solution could include thermal imaging, which seemed complex to implement on a small-scale design. The chosen implementation requires more code to detect false positives. Overall, we feel the final product was a success given that it passed all tests specified within the UAT. Based on the issues discussed here, the system can be greatly improved to avoid slight inconsistencies and false positives.

# References

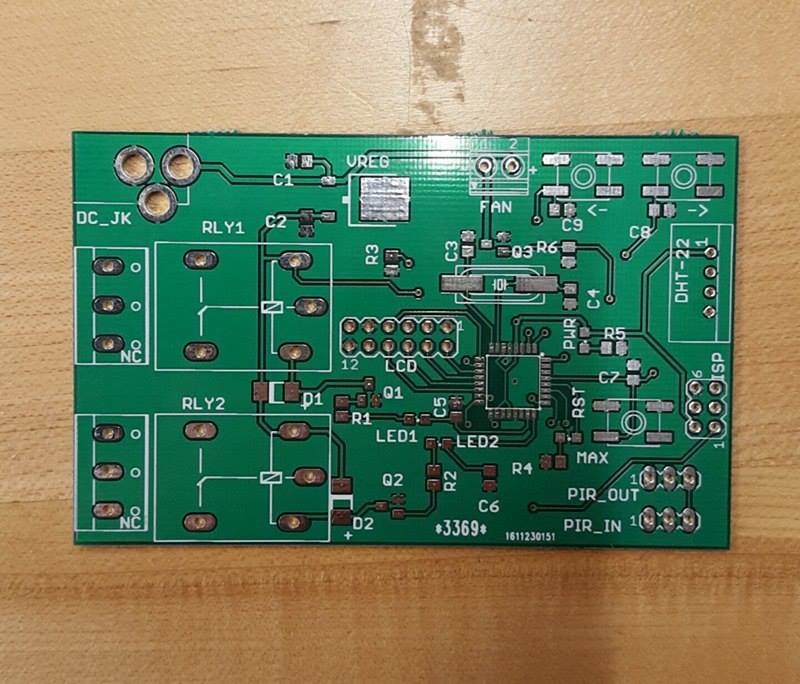
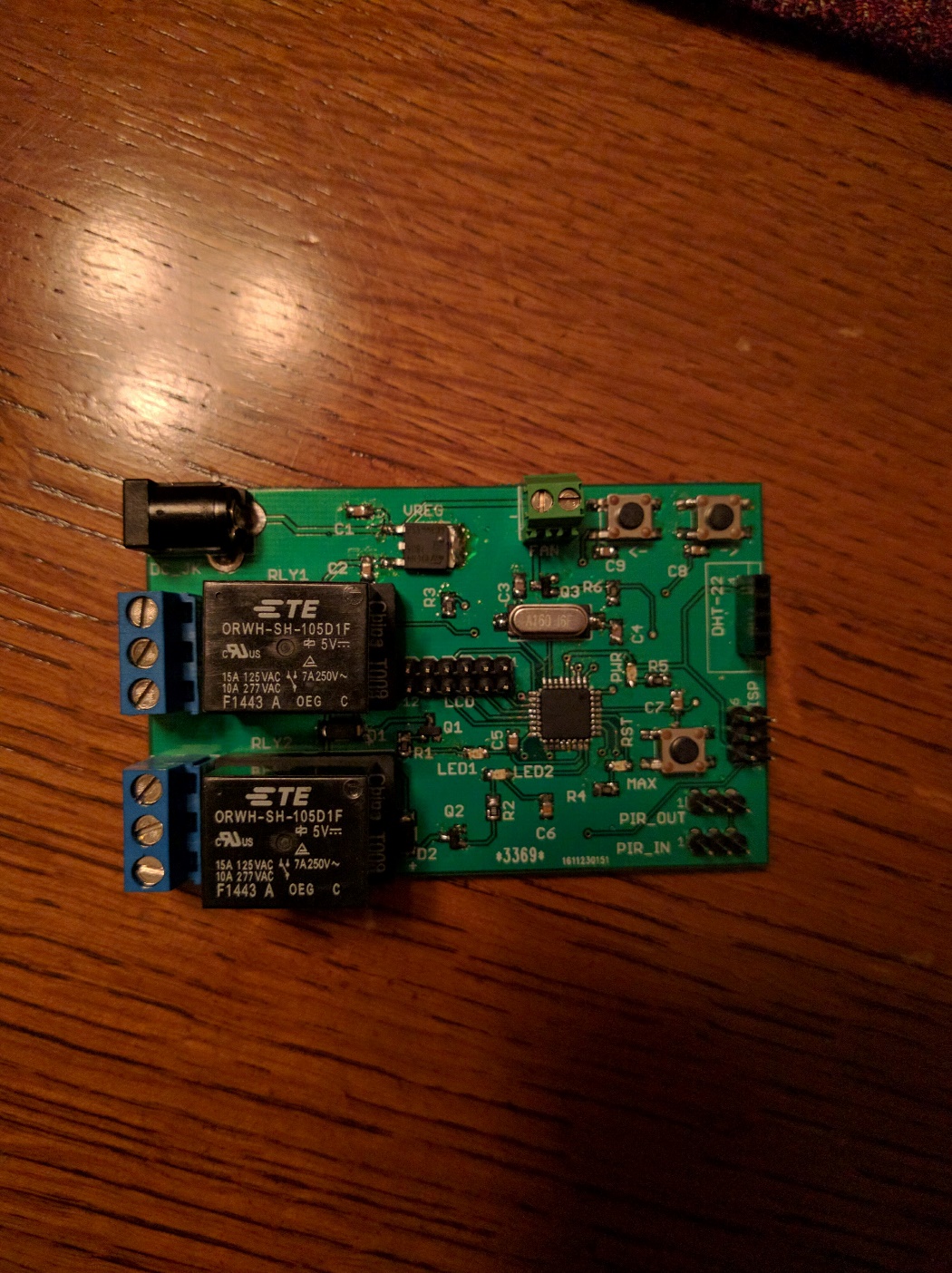
[1] Atmel “8-bit microcontroller with 4/8/16/32Kbytes in-system programmable flash,” ATmega328p datasheet, Dec. 2009 [Revised Oct. 2014].

[2] Jimb0, "Pocket AVR programmer hookup guide," in Sparkfun. [Online]. Available: https://learn.sparkfun.com/tutorials/pocket-avr-programmer-hookup-guide/using-avrdude. Accessed: Dec. 8, 2016.

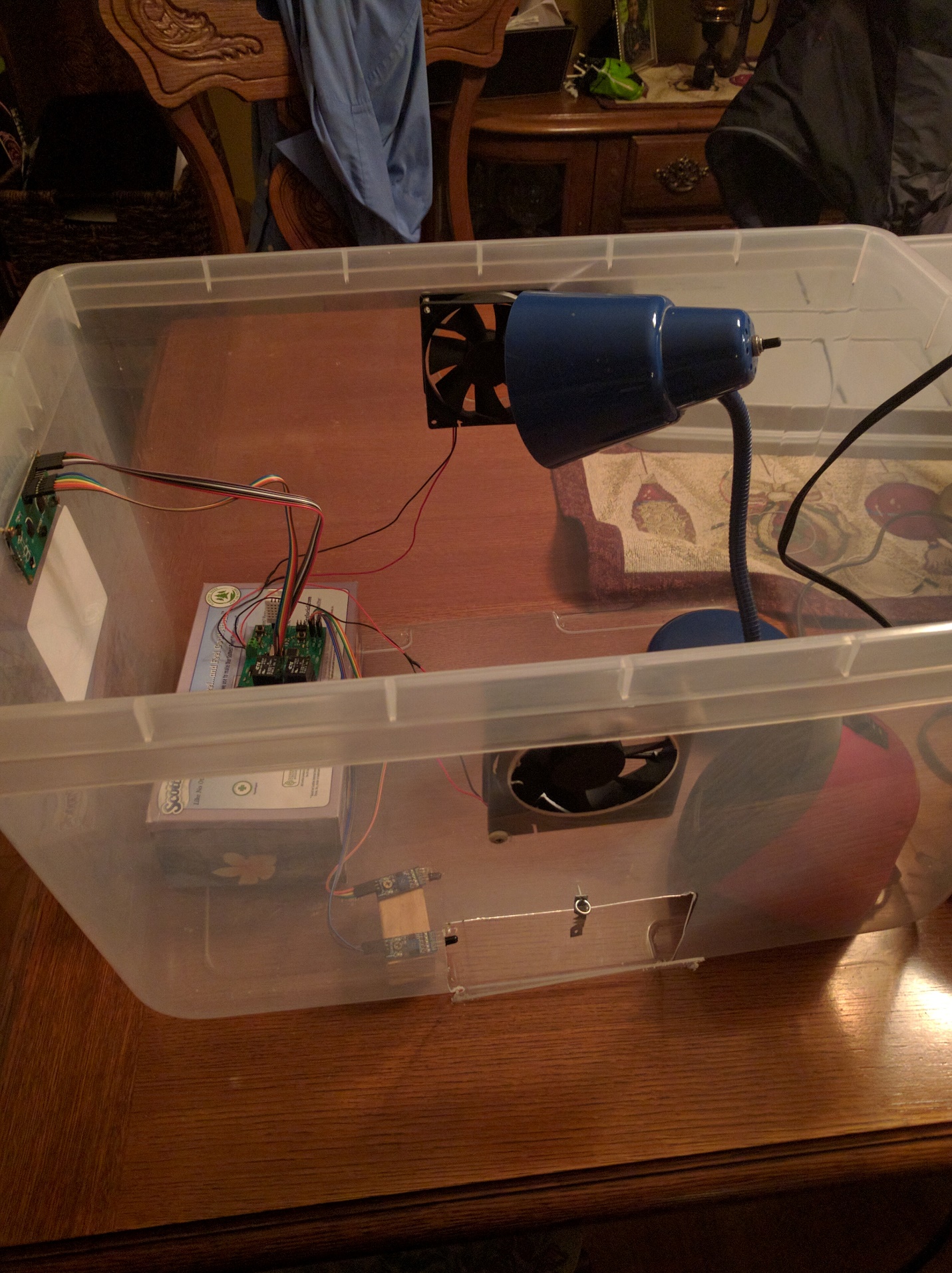
[3] L. George, "Interfacing 16x2 LCD with Atmega32 Microcontroller using Atmel studio," in *electroSome*, electroSome, 2013. [Online]. Available: https://electrosome.com/interfacing-lcd-atmega32-microcontroller-atmel-studio/. Accessed: Nov. 29, 2016.

# Appendix

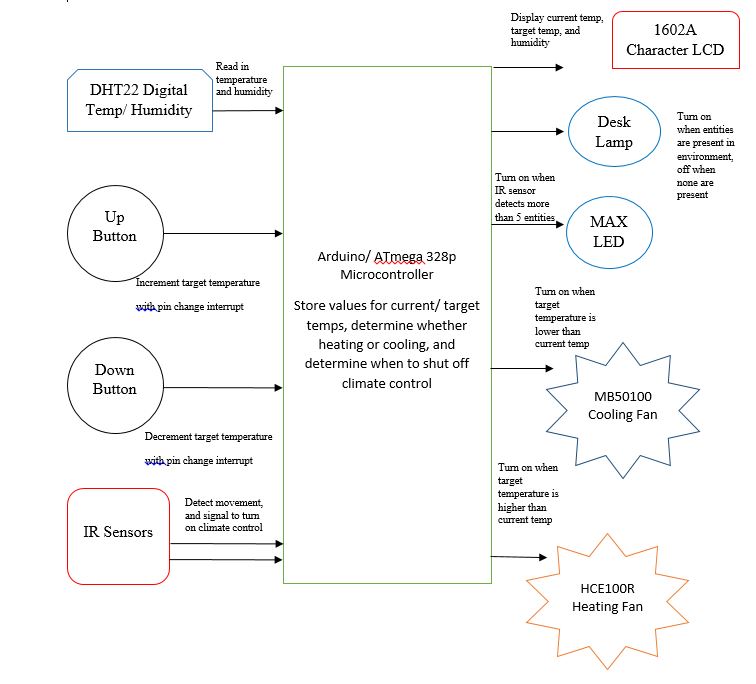
*Figure 1: PCB Before and After Populating*

**

*Figure 2: Modeled Environment for Climate System*

**

*Figure 3: System Flowchart*

****

*Figure 4: Final Firmware*

/\*

\* Climate\_Control.c

\* Version 1

\* Created: 11/19/2016 7:25:49 PM

\* Author : Eric Pires, Richard Harrison, John Lamanuzzi

\*/

#define F\_CPU 16000000UL

#define FAN\_PORT PORTC

#define HEATER\_PIN PORTC1

#define COOLER\_PIN PORTC2

#define IN\_TRIGGER !(PINC & (1<<PINC3))

#define OUT\_TRIGGER !(PINC & (1<<PINC4))

#define SWITCH\_PRESSED !(PIND & (1<<PIND4))

#define MIN\_TEMP 15

#define MAX\_TEMP 30

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "lcd.h"

#include "dht22.h"

volatile float currentTemp = 0;

volatile float setTemp = 0;

volatile float humidity = 0;

volatile int lsIn = 1, lsOut = 1;;

volatile int roomCnt = 0;

// Up Button Interrupt

ISR(INT1\_vect)

{

if(setTemp + 1 <= MAX\_TEMP)

setTemp++;

}

// Down Button Interrupt

ISR(PCINT2\_vect)

{

if(SWITCH\_PRESSED && setTemp -1 >= MIN\_TEMP)

setTemp--;

}

int main(void)

{

// Set LCD Outputs and Initialize

DDRD |= (1<<DDD5) | (1<<DDD6) | (1<<DDD7);

DDRB |= (1<<DDB0) | (1<<DDB1) | (1<<DDB2);

Lcd4\_Clear();

Lcd4\_Init();

Lcd4\_Set\_Cursor(0,1);

Lcd4\_Write\_String("TEMP SET COUNT");

// Initialize Fan Outputs

DDRC |= (1<<DDC1) | (1<<DDC2);

// Initialize Button UI Inputs

DDRD &= ~(1<<DDD3); // Set Up and Down Buttons as Input

DDRD &= ~(1<<DDD4);

PORTD |= (1<<PORTD3) | (1<<PORTD4); // Enable pull-ups

PCMSK2 |= (1<<PCINT19) | (1<<PCINT20); // Mask Button

EIMSK |= (1<<INT1);

PCICR |= (1<<PCIE2); // Enable interrupt bit

EICRA |= (1<<ISC11); // Interrupt generates on falling edge

// Initialize IR Inputs and Outputs

DDRC &= ~(1<<DDC3); // PIR 1 as input

DDRC &= ~(1<<DDC4); // PIR 2 as input

PORTC |= (1<<PORTC3) | (1<<PORTC4); // Enable pull-ups

DDRC |= (1<<DDC5); // MAX as output

// Get current temp to initialize set temp

PORTD |= (1<<PORTD2);

dht\_gettemperature(&setTemp);

// Relay Outputs

DDRC |= (1<<DDC0) | (1<<DDC1);

sei();

while (1)

{

temp\_hum\_disp();

fanControl();

occupancy\_control();

}

}

void temp\_hum\_disp()

{

char buf[3];

Lcd4\_Set\_Cursor(2,0);

dht\_gettemperaturehumidity(&currentTemp,&humidity);

*dtostrf*(currentTemp,0,0,buf);

Lcd4\_Write\_String(buf);

Lcd4\_Write\_String(" ");

*dtostrf*(setTemp,0,0,buf);

Lcd4\_Write\_String(buf);

Lcd4\_Write\_String(" ");

*dtostrf*(humidity,0,0,buf);

//Lcd4\_Write\_String(buf);

}

void fanControl()

{

currentTemp = *trunc*(currentTemp);

setTemp = *trunc*(setTemp);

// Cooler is activated, heater is deactivated (Cools room)

if(currentTemp > setTemp)

{

FAN\_PORT &= ~(1 << HEATER\_PIN);

FAN\_PORT |= (1 << COOLER\_PIN);

}

// Heater is activated, cooler is deactivated (Heats room)

else if(currentTemp < setTemp)

{

FAN\_PORT &= ~(1 << COOLER\_PIN);

FAN\_PORT |= (1 << HEATER\_PIN);

}

// Heater and cooler deactivated (Temperature matches user input)

else

{

FAN\_PORT &= ~(1 << COOLER\_PIN);

FAN\_PORT &= ~(1 << HEATER\_PIN);

}

}

void roomIn()

{

int csIn = IN\_TRIGGER;

// Compare current state to last state

if(csIn != lsIn)

{

if(IN\_TRIGGER)

roomCnt++;

*\_delay\_ms*(500);

}

// Save previous state

lsIn = csIn;

}

void roomOut()

{

int csOut = OUT\_TRIGGER;

if(csOut != lsOut)

{

if(OUT\_TRIGGER && roomCnt > 0)

roomCnt--;

*\_delay\_ms*(500);

}

lsOut = csOut;

}

void occupancy\_control()

{

char buf[3];

roomIn();

roomOut();

// MAX LED control

if(roomCnt > 2)

// MAX LED set HIGH

PORTC |= (1<<PORTC5);

else

PORTC &= ~(1<<PORTC5);

// Light Relay Control

if (roomCnt > 0)

// Light relay set HIGH

PORTC |= (1<<PORTC0);

else

// Lights set LOW

PORTC &= ~(1<<PORTC0);

Lcd4\_Set\_Cursor(2,14);

*itoa*(roomCnt,buf,10);

Lcd4\_Write\_String(buf);

}