**Project Lab 2**

**Climate Control and Room Occupancy System Fabrication**

Eric Pires

John Lamanuzzi

Richard Harrison

ECE 388

Team 3

Lab Date: November 9, 2016

Due Date: November 30, 2016



**Table of Contents**

[Introduction 2](#_Toc468223498)

[Methods and Testing Procedures 2](#_Toc468223499)

[Board Design Finalization 2](#_Toc468223500)

[Firmware Design 3](#_Toc468223501)

[Results 6](#_Toc468223502)

[Board Design Finalization 6](#_Toc468223503)

[Discussion 10](#_Toc468223504)

[Conclusion 11](#_Toc468223505)

[Reflection 12](#_Toc468223506)

[References 13](#_Toc468223507)

[Appendix 14](#_Toc468223508)

Abstract

The goal of this laboratory was to finalize the PCB fabrication and driver software in preparation of the final due date. Gerber files were examined and modified to improve on the previous PCB design. The Acceptance Test Plan (ATP) was also created to determine the overall functionality of the System Under Test (SUT).

# Introduction

The objective of this lab was to finalize the fabrication of the climate and room occupancy control system. Due to the time constraints of the shipping process, the firmware was developed and debugged on the ATmega328p Xplained Board. The following software tools were used in the fabrication process.

* Atmel Studio 7
* EAGLE
* Gerbv

The firmware was written and debugged inside Atmel Studio. EAGLE and Gerbv was used to modify the gerber files for board design.

# Methods and Testing Procedures

## Board Design Finalization

The first phase of the laboratory was to improve on the PCB design. Using the gerbv software, the gerber files were carefully observed and modified. While viewing the gerber files, a major issue was detected. The LCD VO pin was connected to a pull-down resistor. The pin required a pull-up resistor. This issue was easily resolved by adjusting the schematic and board inside EAGLE and processing the gerbers once more. The gerbers were constantly modified to fix traces and silkscreen issues. These issues were determined through careful inspection of the gerbv project file.

## Firmware Design

The following phase of the laboratory consisted of software design. The software written for the prototyping phase was done in the Arduino environment. Since this environment runs on C++, the code required modification to be ported to Atmel Studio. The software was written for each individual module and then combined to one file. These modules will be discussed below.

LCD

The first module tested was the LCD. Since the display module contains a ST7066U-0A driver, code for this specific driver was researched and utilized for this system. This was done by including the header and c file responsible for writing to the LCD. These files can be found in Figure 1 of the Appendix. The code provides functions for initializing, clearing, and writing to the LCD. These routines help abstract the operation of the LCD and help simplify the main code of the system. The code was edited to include the system requirements only. This meant removing the 8-bit data transfer functions.

DHT-22

The next module was the DHT-22 temperature and humidity sensor. Research was done to locate a working library to be interfaced with the system’s microcontroller. The code was imported into the system project file. Similar to the first module, the DHT header was edited to only contain the system requirements. This code can be found in Figure 2.

Temperature Control

The next module was the temperature control code. Simply put, the code toggled the given output pin based on a conditional statement. Whenever the set temperature was less than the currently measured temperature, the heater pin was enabled to switch on the relay. When the set temperature exceeded the currently measured temperature, the cooler pin was enabled to switch on the cooling fans. The following flowchart demonstrates this functionality.



Since the DHT-22 is the slowest responding sensor, such data is not time critical. Therefore, this operation is polled inside the main control loop. The user input, on the other hand, is time critical and is handled through an interrupt service routine. Therefore, the set temperature is a higher precedence than the currently read temperature. This ensures that whenever the DHT reads a new value, it will already have a set value to compare itself to. The user input diagram can be found below. This code can be found in Figure 3.



Occupancy Control

The final module was the occupancy control. Whenever the system detected occupancy in the environment, the light pin was enabled to switch on the relay for the desk lamp. In accordance to this, the number of occupants in the environment was checked with the maximum defined value. If this number was exceeded, then the MAX led pin would be enabled. This code can be found in Figure 4. The flowchart for this operation can be found below.

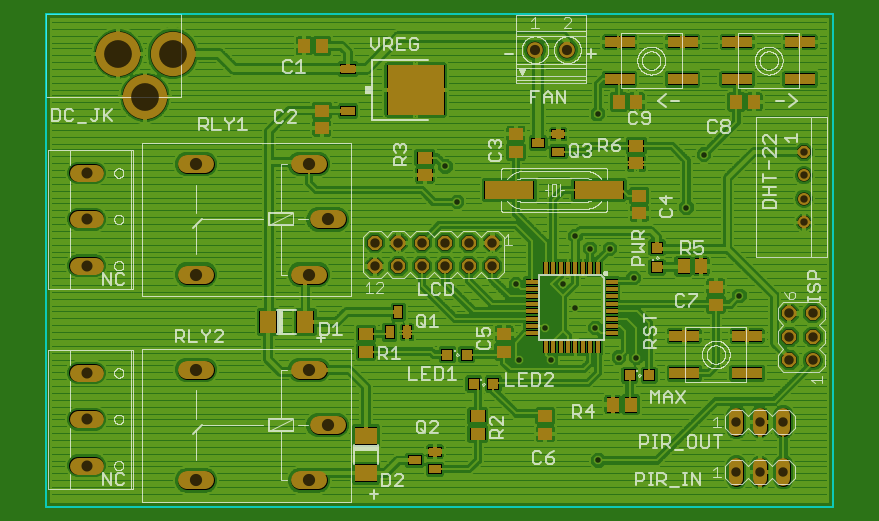


After each module was written, the Xplained Mini was loaded with the given code and used to control each device in test. Repeating this prototyping phase ensured that the software size would not exceed the maximum chip memory (32KB) as well as the time constraints for the user input.

# Results

## Board Design Finalization

The final design layout was packaged and sent to DirtyPCBs for fabrication. The board can be found below.



The board will be delivered soon and properly populated upon its arrival. It will then undergo the tests found in the ATP attached to this report.

Firmware Design

The developed firmware was tested using a simple test matrix. This was designed to ensure that the driver provided basic functionality to the given module. The following testing matrix was used for the LCD.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Name | Testing Question | Procedure Description | Result |
| 1 | Write Check | Does the firmware initialize and write to the LCD | Send the LCD a simple message by initializing and adjusting the device to a given cursor location | **Pass** |
| 2 | Counter | Does the system respond accordingly to a user input (Push Button)? | Use the button to adjust the counter value displayed on the LCD. Observe to see if each press responds accordingly to the new value. | **Acceptable**  No internal debouncing is present within the used button. Value increases twice the amount of presses. |
| 3 | Clear Check | Does the system initialize to the default settings upon reset? | After adjusting the counter, reset the microcontroller and observe the new counter value. | **Pass** |

The second test received an acceptable result because it contained no debouncing solution. This issue has been accounted for in the board design and can thus be negligible. Despite this, all tests passed after the library code was edited for this particular microcontroller.

The DHT functionality was tested through the usage of the LCD. The temperature and humidity values were printed and examined on the display. The following testing matrix was used to determine its functionality.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Name | Testing Question | Procedure Description | Result |
| 1 | Read / Write Temperature | Does the firmware recognize the input data and print it out onto the LCD? | Read the DHT temperature and pass it to the LCD using the write function. Compare with the actual room temperature to validate DHT data by averaging ten sample readings. | **Pass**  Average DHT Temperature: 19.5 C  Measured Room Temperature: 20 C  (Based on central thermostat reading) |
| 2 | Read / Write Humidity | Does the firmware recognize the input data and print it out onto the LCD? | Read the DHT humidity and pass it to the LCD. Compare with the actual room humidity to validate the data by averaging ten sample readings | **Pass**  Average DHT Humidity: 42.2%  Measured Room Humidity: 40 C  (Based on humidity sensor reading) |
| 3 | Clear Check | Does the system initialize to the default settings upon reset? | After adjusting the counter, reset the microcontroller and observe the new counter value. | **Pass** |

All three tests passed with expected readings. By comparing the reading to an existing temperature and humidity sensor, the results were confirmed to be accurate. Likewise, the reinitialization of the counter data was confirmed to be reset after the system was rebooted. The code was still present within the internal memory and the temperature data initialized to the specified default value. In the final application, this value will initialize to be the currently measured temperature. This ensures that the system will not trigger either fan upon boot up.

The next module was the temperature and occupancy control. This was tested by giving the system certain inputs and observing the outcome. The temperature control involved adjusting the set temperature to observe the cooler and heater response. The occupancy control involved reading the IR sensor inputs and observing the system output. The testing matrix can be found below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Number | Test Name | Testing Question | Procedure Description | Result |
| 1 | Heater Test | Does the heater activate when the user inputs a higher temperature than the currently read temperature? | Adjust the set temperature value to exceed the room temperature value. Ensure that the device is properly activated and powered. | **Pass**  The heater is powered with the relay switched 110V signal, drawing around 1.5A. |
| 2 | Cooler Test | Does the cooler activate when the user inputs a lower temperature than the currently read temperature? | Adjust the set temperature value to be less than the room temperature value. Ensure that the device is properly activated and powered. | **Pass**  The cooler is powered with the logic signal. The measured current is 190mA. |
| 3 | Entrance Check | Does the system detect occupancy within the enclosed environment? | Trigger the entrance IR sensor and observe the lamp response. Ensure the lamp is properly activated and powered | **Acceptable**  The system responds to the pin change induced by the IR sensor, however the ISR counter is not properly debounced. |
| 4 | Maximum Occupancy Check | Does the system respond when the maximum number of occupants has been detected? | Trigger the entrance IR to reach over the maximum threshold. Ensure the MAX led responds to the give input. | **Acceptable**  (See above) |
| 5 | Exit Check | Does the system detect when all occupants exit the enclosed environment | Trigger the exit IR sensor to observe the lamp response. Ensure the lamp is properly activated and powered. | **Acceptable**  (See above) |

The IR sensor drivers successfully calls upon the ISR when motion is detected. However, the routine is not properly debounced, producing an inaccurate counter for the number of times the sensor activates. Possible adjustments are mentioned in the Discussion.

# Discussion

The firmware development for this system proved to be more challenging than previously expected. Without the convenience of the Arduino environment, the basic functionality of the GPIO registers was researched. Such functions like digitalRead and digitalWrite were disregarded and replaced with bit manipulating macros. For example, to read a pin, the following macro was utilized:

#define CHECKBIT(ADDRESS,BIT) (ADDRESS & (1<<BIT))

Using such techniques requires more knowledge on the registers and their bits. This macro was found in the AVR forum for introductory programming [1]. The basic register and GPIO functionality was found in the datasheet for the ATmega328p [2]. The most challenging task was to port the LCD and DHT libraries over to a C project. Fortunately, two basic libraries were found and included into the project. These libraries were not written for the ATmega328p, but were modified to support it. In addition, the functions and definitions that were not utilized were deleted to save space on the internal memory. Implementing these libraries by scratch might have simplified the process, but this method seemed time consuming and complex [3] [4]. The drivers are mostly ready and can be imported to the system upon arrival. However, the occupancy control still requires some minor modifications. The debouncing solution created in the Arduino environment did not port well to the C environment. The following week will be dedicated to resolving this issue for the final product. The initially specified testing requirements state that the occupancy control will not be fully accurate, but it its within our intention to make it as accurate as possible. In addition, the user interface will be evaluated once more. The firmware only contains code for one button, since there is only one button on the development board. Creating a second routine will be easily accomplished and tested upon the arrival of the boards.

# Conclusion

In conclusion, the laboratory further guided the final stages of the system development. The board design had its final optimizations and downsizings and were submitted for fabrication during this project lab. Although the boards have not been received in time for testing, this allowed for more focus on preparing the firmware for development of the product. For the firmware, this lab focused on moving away from the Arduino environment to development in Atmel Studio. Arduino’s abstraction was helpful in the beginning steps of the design project. However, reducing code size to fit the ATMega328P’s storage and more traditional (and familiar) C programming fell in favor, bringing about new requirements for the firmware development. These requirements were converting the Arduino libraries to C code, which the team researched and either developed new code or adapted existing code. With the combination of the finalized board design and firmware developed for this finalized design, the team can move forward with the final steps of the design project, which is to ensure the working state of the system as a whole using the newly updated firmware and a model environment for testing.

# Reflection

The prolonged PCB development phase prevented delivery of the finished project boards for preliminary testing with the firmware that was developed; Although, the board design was completed and submitted on time. In the meantime, the core firmware was developed and compiled onto the developer board. The button code will be modified to match the interrupt pins of the two input buttons. Furthermore, more research will be done regarding a reset interrupt to allow a system reset without disconnecting the power source. When the final project boards are delivered, the firmware will be once more evaluated and modified.

The project’s firmware was the main focus of this lab, as well as finalizing of the EAGLE designs for submission. The goal of firmware development for this lab is the conversion of code from the previously used Arduino libraries to use with Atmel Studio and avr-gcc. The advantage for using Atmel Studio and avr-gcc over Arduino is that while Arduino’s C++ code is abstracted (and therefore can be easier to use), it increases code size, which can be a constraint with the ATMega328P’s anemic on-chip storage. The use of Atmel Studio is also conducive to loading commercial real time operating systems. The Arduino platform is also inefficient when dealing with interrupt based routines. The AVR platform is much better equipped to handle such applications.

The main consideration when converting code from Arduino to Atmel is understanding the helpful abstraction that Arduino libraries offer. Researching these operations helps provide a better understanding of the SUT. Fortunately, the Atmel community contains many helpful tutorials and libraries to help aid in system development, much like the Arduino community.

# References

[1] E. Weddington, "Bit manipulation (AKA ‘Programming 101’)," in *AVR Freaks*, 2006. [Online]. Available: http://www.avrfreaks.net/forum/tut-c-bit-manipulation-aka-programming-101?page=all. Accessed: Nov. 29, 2016.

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

[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.

[4] D. Gironi, "Reading temperature and humidity on AVR…," in Blogspot, 2014. [Online]. Available:http://davidegironi.blogspot.com/2013/02/reading-temperature-and-humidity-on-avr.html?m=1. Accessed: Nov. 29, 2016.

# Appendix

*Figure 1: LCD Code*

// Set LCD Outputs and Initialize

DDRD |= (1<<DDD5);

DDRD |= (1<<DDD6);

DDRD |= (1<<DDD7);

DDRB |= (1<<DDB0);

DDRB |= (1<<DDB1);

DDRB |= (1<<DDB2);

Lcd4\_Clear();

Lcd4\_Init();

Lcd4\_Set\_Cursor(0,1);

Lcd4\_Write\_String("TEMP SET HUM%");

*Figure 2: DHT Code*

*void temp\_hum\_disp()*

*{*

*char buf[3];*

*float humidity = 0;*

*Lcd4\_Set\_Cursor(2,0);*

*dht\_gettemperaturehumidity(&currentTemp,&humidity);*

*dtostrf(trunc(currentTemp),1,1,buf);*

*Lcd4\_Write\_String(buf);*

*Lcd4\_Write\_String(" ");*

*itoa(setTemp,buf,10);*

*Lcd4\_Write\_String(buf);*

*Lcd4\_Write\_String(" ");*

*dtostrf(trunc(humidity),1,1,buf);*

*Lcd4\_Write\_String(buf);*

*}*

*Figure 3A: DHT control*

*void fanControl()*

*{*

*// 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);*

*}*

*}*

*Figure 3B: Button Interface*

ISR(PCINT0\_vect){

setTemp++;

}

// In main function

DDRB &= ~(1<<DDB7); // Set PB7 as Input

PORTB |= (1<<PORTB7); // Enable PB7 pull-up

PCMSK0 |= (1<<PCINT7); // Mask Button

PCICR |= (1<<PCIE0); // Enable interrupt bit

sei();

*Figure 4: Occupancy Control*

ISR(PCINT1\_vect)

{

if(IN\_TRIGGER)

{ // Check for Falling Edge

if(!lsIn)

{

roomCnt++;

lsIn = 0; // Toggles lsIn low

}

}

else

{ // Check for Rising Edge

if (lsIn)

lsIn = 1;

}

if(roomCnt > 5)

PORTC |= (1<<PORTC5);

else

PORTC &= ~(1<<PORTC5);

\_delay\_us(350);

}