State University of New York at New Paltz
Department of Electrical and Computer Engineering

Embedded Systems Final Project
EGC-416

Design of a heating and cooling system with thermostatic control

Group Members	Department	Major Contribution
Paul Boston	CE	Theory, Design, Software, Hardware
Mitchell Wagner	CE	Theory, Design, Software, Hardware

Course Professor: Mike Otis

Due: 12/15/14

Abstract:

Most every home has some sort of heating and cooling system. Often times a person will desire to change the temperature that it is to a temperature that will be desired, as well as have it remain at that temperature with not much varying. These are important aspects when creating a house system.

Table of Contents:

Introduction

Heating and cooling are a major part of any home system. For any house it is important for the user that the desired temperature be set to the desired and that it will remain at that temperature and be able to compensate if the actual temp is to hot or cool. Important factors when using are the efficiently how much power, the amount noise it creates and how big this system is. The 3 main parts of this lab is the regulating the temperature, using 2 processors and displaying data.

Regulating the temperature had three parts, heating, cooling, and inputting the current temperature. Both heating and cooling involved using fans to blow the air around to change the temperature. Heating was done through using a coil system that would be heated up electronically. The cooling was done by having a copper wire attached to the outside of this box and the fan would blow the new temp through the house. This can be seen in Figure 1 all the parts of the system.

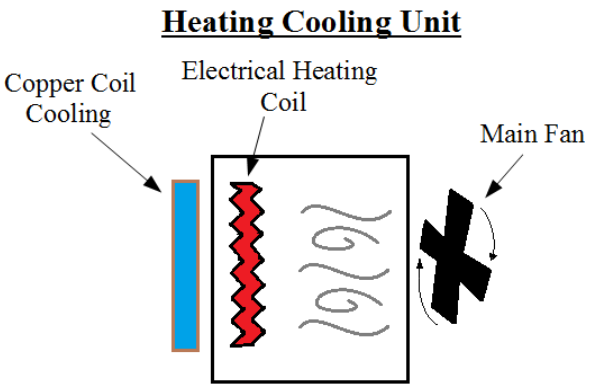


Figure 1: Heating, Cooling Unit

The use of two processor was done so that one was devoted to the regulation of temperature. The second processor was used so that the user could see the temperature that was being desired and the current temperature. This processor would involve using a Hex keypad and LCD screen to display the temperatures. The data would be sent from each processor regularly through the use of the UART transmitting and receiving.

The processor requires many types of programming UART, ADC conversions, Interrupts, timers, inputs and outputs. All this was coded and compiled in C programming language.

Design Software

This project is comprised of 2 processors with a multitude of hardware attached. In order to get this accomplished we needed to utilize a couple of onboard features of the microcontrollers such as interrupts, timers, UART and ADC. With all of these onboard features we were able to create our design to control our hardware.

In our first processor (or processor1), its job was to deal with user input and output. Its job is solely to display the current temperature, display the set temperature and read in a new set temperature. It would then send the set temperature to the other processor and read in the current temperature from the other processor all via UART. In this processor we used an interrupt function and UART to control our data (figure 1). We also needed a hex keypad and LCD screen, but we already knew how to program those from microprocessors last semester.

The interrupts job in this code is when an external button is pressed; the code inside the main loop is halted and executes a separate set of code only executed when an interrupt occurs. In this code, we used it to control when to update the new set temperature number. When the button is pressed, the system stops and asks you on the LCD screen to input a new number via the hex keypad, this number is then read into the system and set as an integer to be sent over UART to the other processor and also to be displayed on the screen. To use the hex keypad, we have to call its function that we had to create; this function job is to read which column the button being pressed is in and then read the row that the button being pressed is in. By doing that now we have an exact location of where the button is allowing us to assign it the number that it represents, we then return that number.

This processor communicates to the other processor via UART, UART stands for Universal Asynchronous Receiver/Transmitter. So it uses 2 pins Tx and Rx. How this would work is you would hook up Tx pin of one processor to the Rx pin of the other processor and vice versa. Then in our code we specified that first we needed to initialize the UART startup. Then we needed to check to make sure the line is clear before we start to send data. Then we send our data to the UART Tx pin in the form of an 8 bit int value. After that we need to wait for the line to be clear again, and our bit was sent.

In processor 1 we utilized these functions to make this processor work accordingly, in the appendix you can find all of the code we created to make all of this work. In there you will find other functions that we used such as delay loops, startup initializations, and data writes and command writes.

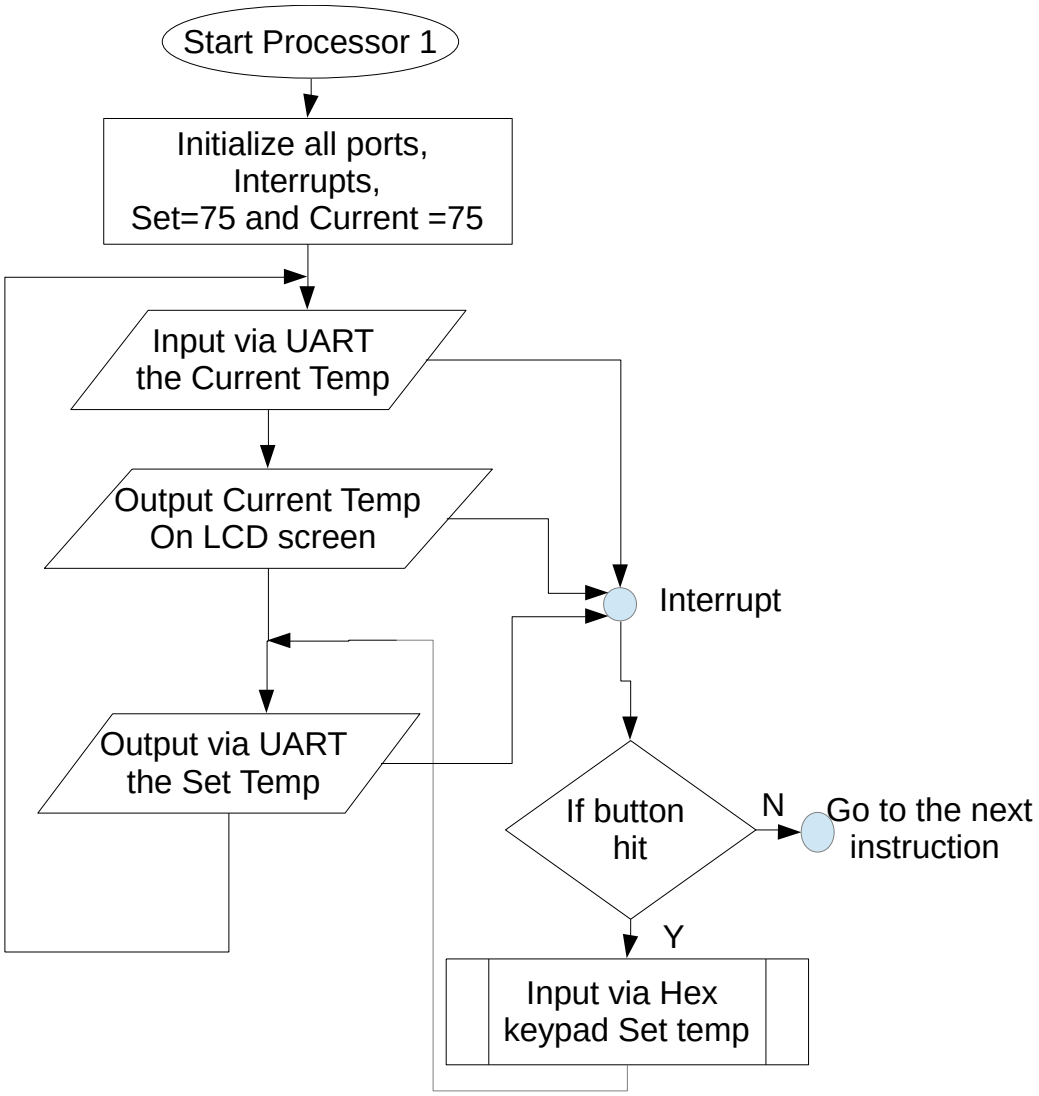


Figure 1

In our second processor (or processor 2), its job is to deal with all of the heating and cooling equipment as well as read the temperature. This microcontroller will read the temperature periodically via a timer interrupt, and then it will receive the set temperature from the other processor and determine how to set the heating and cooling hardware (figure 2).

Connected to this processor is a temperature sensor, the processor is able to read this value in via an ADC converter. ADC means Analog to Digital Converter. The signal coming from the temperature sensor needs to be converted from analog state (0-3.3V) to digital 12 bit int. The number that the processor converts it to is a unit of steps. How the processor understands this data is by looking at it and figuring out what the percentage is of the voltage coming in to the max voltage it can read, max voltage being 65536 (max number of steps). Once a little bit of math is done we can say Current = input * (330.0 / 65536). The 330 is because the temperature sensor works by increase 10mV per degree (C) so we needed to convert over to degrees.

The interrupt in this processor works a little differently than in the other processor. In the other processor we needed to push an external button when we wanted to interrupt, but in this processor we want it to do it automatically based off of the system clock. Every certain amount of system cycles, or every time the timer buffer fills up to the top, the system will execute an interrupt. Just as before when an interrupt is triggered the processor stops what it is doing in the main loop and starts what is in the interrupt function. In this case the ADC conversion, if statements that control the heating and cooling hardware and the UART transmit and receive are located here. In the main loop all we have is a delay loop to control how often the timer is done.

In the main loop, we have the if statements to control our hardware. These if statements basically say that if the temperature is less than the set, then turn on the heat; and if the temperature is greater then set then turn on the cooling. Unlike a simple bang bang controller, are system self corrects before problems occurs. When we are heating and between the set temperature and 3 degrees below, we start to cool in order to self correct or any overshoot. The same is not done when we are cooling because of the fact that the cooling process is so slow that we can't overshoot if we tried with our current hardware. During all of these stages the fan always remains on to provide circulation to the house. Lastly if the temperature is equal to set the processor turns all the components off.

In this processor the UART functions are all the same. Rx and Tx commands, functions and initializations are all exactly the same from the other processor. In processor 2 we utilized these functions to make this processor work accordingly, in the appendix you can find all of the code we created to make all of this work.

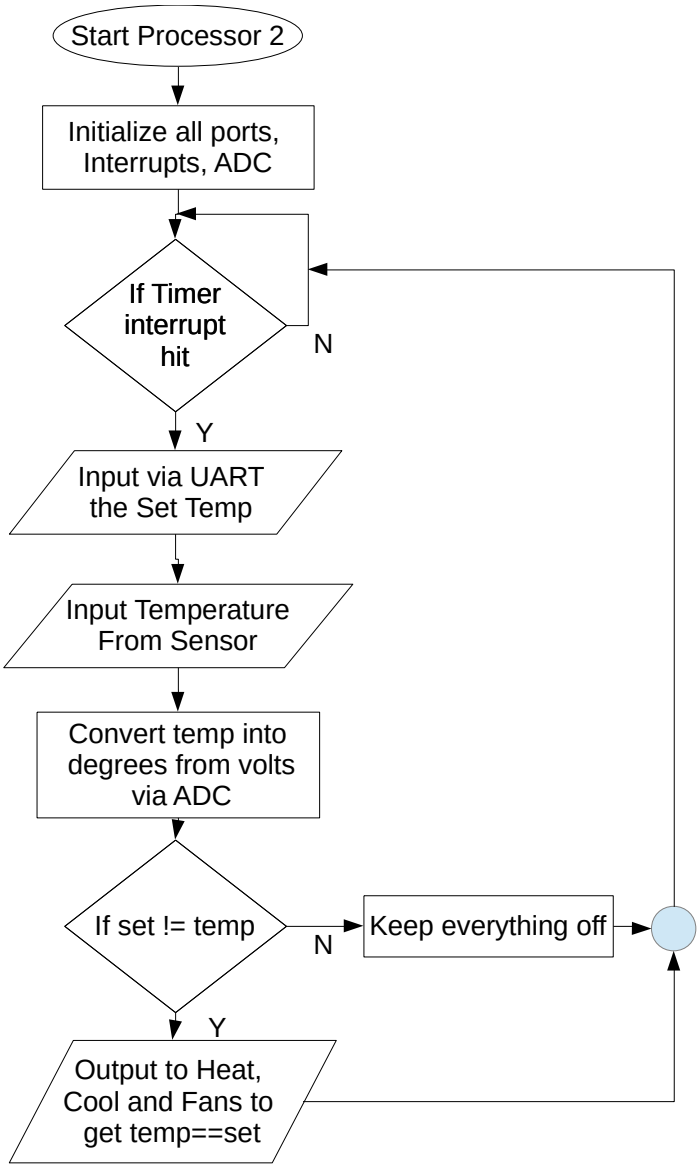
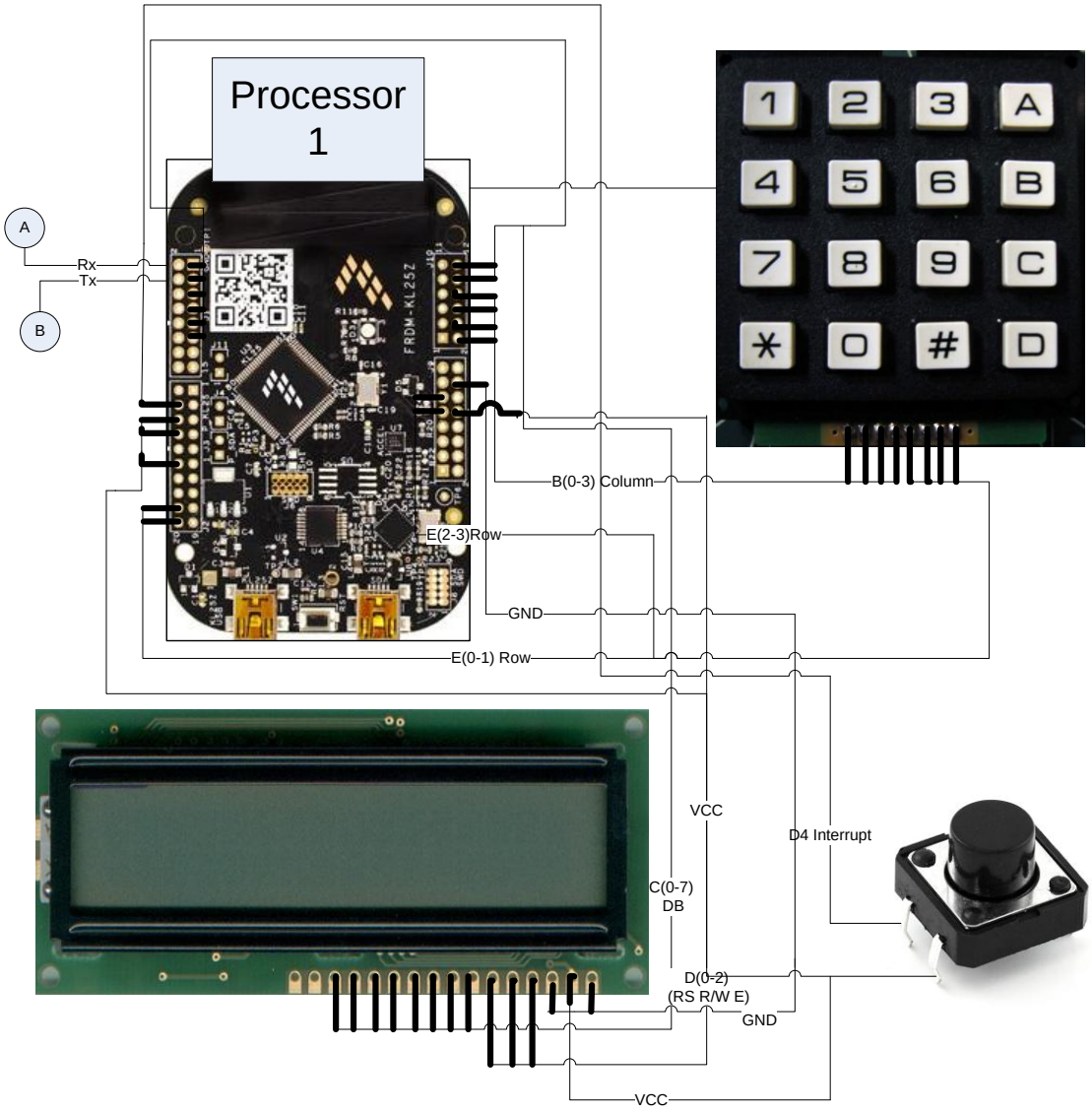

Start Processor 2
Initialize all ports, Interrupts, ADC
If Timer interrupt hit
N
Y
Input via UART the Set Temp
Input Temperature From Sensor
Convert temp into degrees from volts via ADC
If set != temp
N
Keep everything off
Y
Output to Heat, Cool and Fans to get temp==set

Figure 2

Design Hardware

Processor 1:

Processor 1 involved using external hardware of a hex keypad, LCD screen and a button. The hex keypad was used as an input utilizing Ports B and E for the Hex keypad, Port B represented the columns and Port E would represent the rows. The LCD screen would use Port C and Port D. UART is used through Port A.



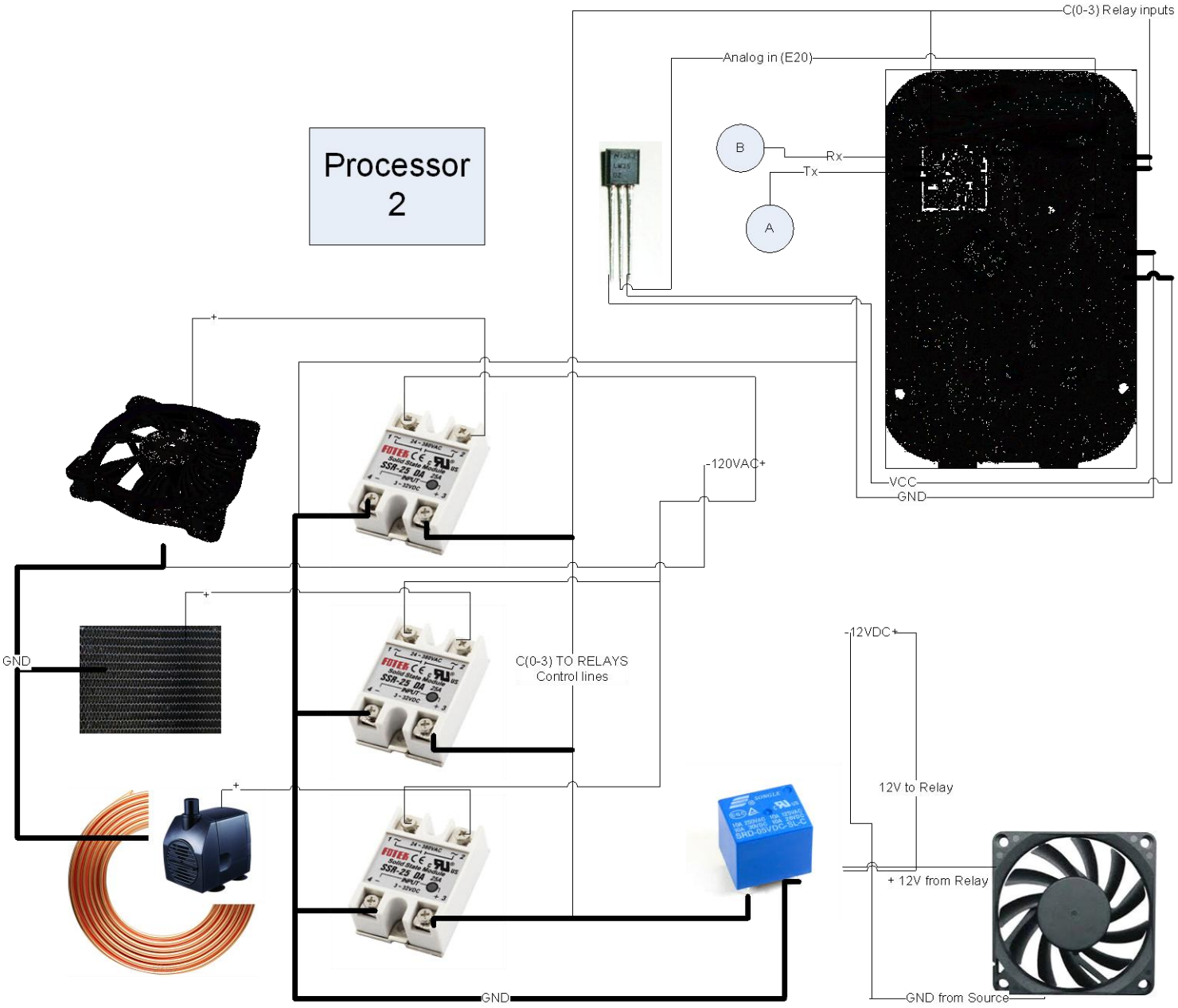
Processor 2:

Processor 2 involved much more hardware than the other processor. For this processor we used a temperature sensor LM35 as well as the heating and cooling units, 2 fans, one the main and the other an exhaust fan. Solid state relays SSR and batteries were used.

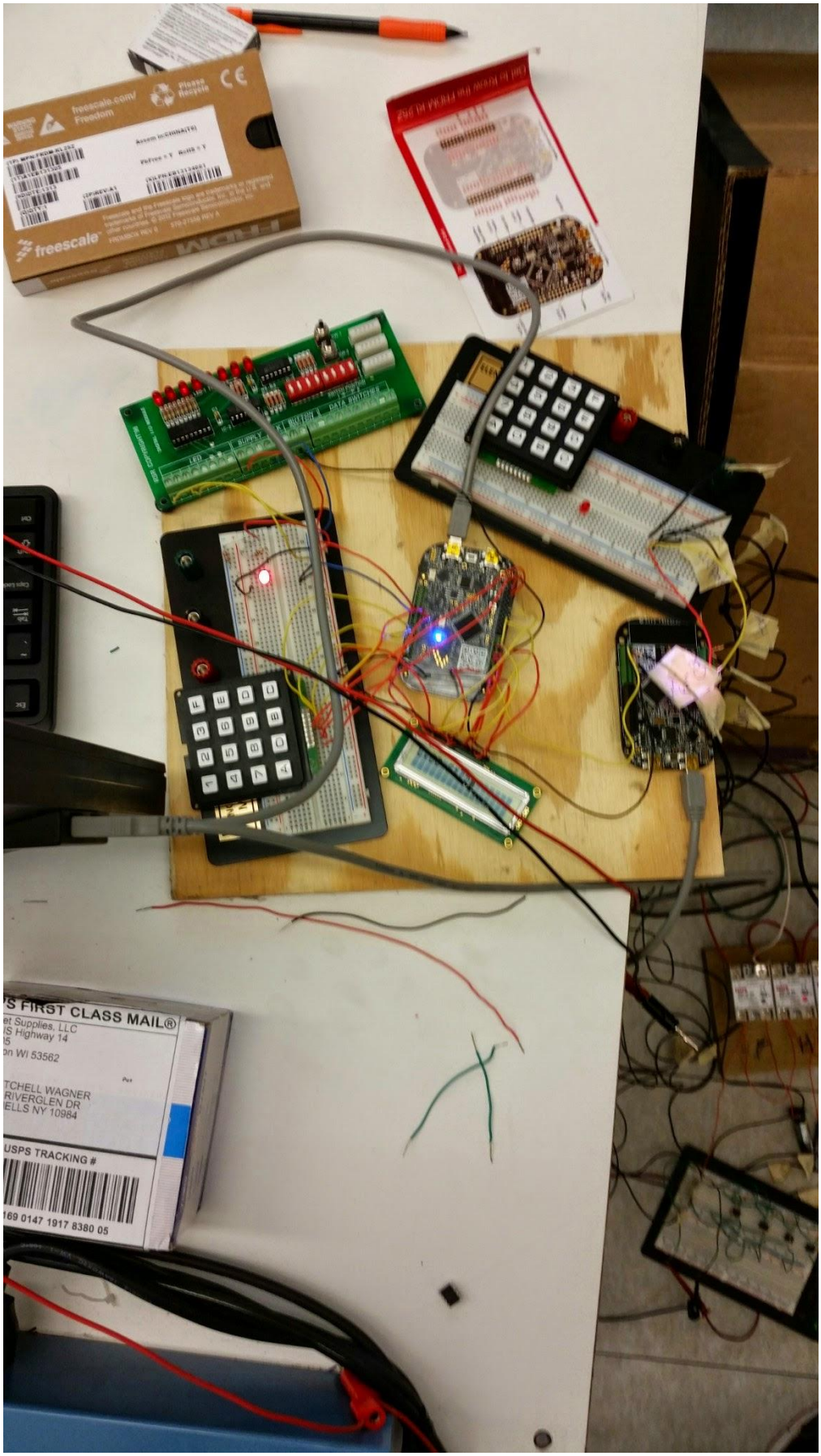
The temperature sensor LM35 was reading the temperature of the house and displays the data in voltage this needed to be converted to the proper units of degree. This sensor would be constantly need to be updated to tell the processors what systems needed to be on.

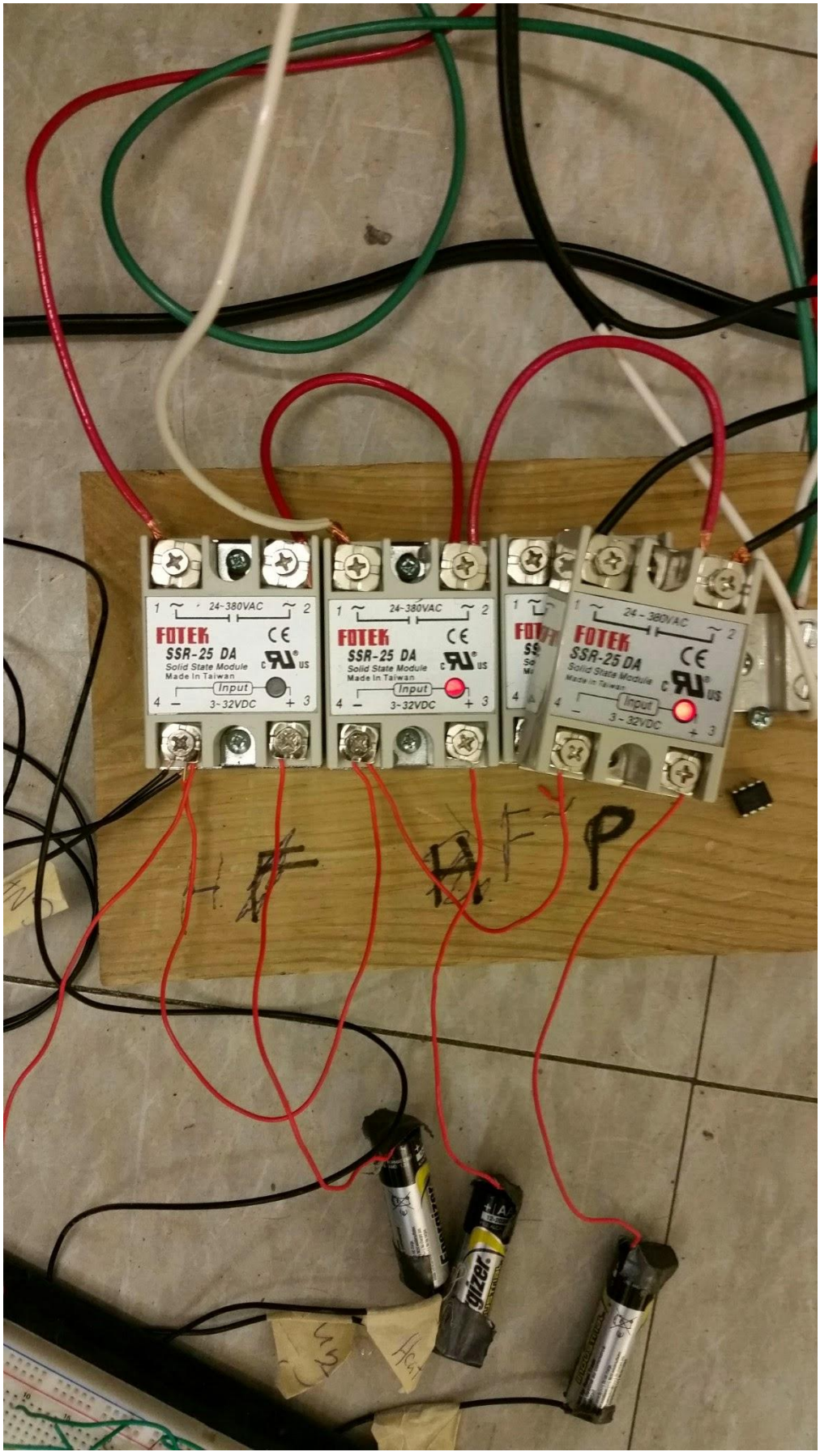
The heating unit involved a coil that would be heated electronically a fan on the other side would blow the warm air through the house as the temperature started to rise and it was within 3 degrees of the desired temperature the cooling unit would turn on to steady the temperature so there would be no overshoot. The cooling process was accomplished by a having a pump take ice cold water through a tub in to a copper coiled in front of the box that held the fan and the heating coil. When the cooling cycle was on the main fan would be on to circulate the cool air. The exhaust fan or attic fan would be one only if the temperature of the house was 3 degrees higher than the set temperature. This is because the temperature needed to go down much faster than.

The batteries and solid state relays were used to activate all the hardware components of the house. Since the different hardware required more voltage the relays were used to when the input voltage was hit this would use the increased voltage be passed through. The relays required a increase amount of current this was tried through the use amplifiers but since it wasn't very successful the plan was shifted into using batteries to increase the voltage to activate the different relays.


Processor 2
C(0-3) Relay inputs
Analog in (E20)
B
Rx
Tx
A
VCC
GND
120VAC+
+
GND
C(0-3) TO RELAYS Control lines
12VDC+
12V to Relay
+ 12V from Relay
GND from Source
GND


CORROSIVE
8
ULTRA CLUB


freescale
FIRST CLASS MAIL®
USPS TRACKING #


FOTEK
SSR-25 DA
Solid State Module
Made In Taiwan
24~380VAC
Input
3~32VDC

Conclusion

Heating and cooling are a big part of a home system. The accomplishment of creating a heat and cooling system show the ability to create much more creative and complex coding and understanding of the processor. Being able to make us of different interrupts, UART, ADC, and inputting from different devices are key aspects of any software and hardware experience.

Processor had several areas of difficulty or problems that required rethinking and some points redesign the system. One such area involved when using amplifiers to be able to increase the signal to the different relays. It was seem that the voltage would be correct but very likely the amperage was inaccurate. Several ideas of solutions to this problem would involve the use of voltage buffers or the design change to our project was the used of batteries. The batteries were used to control amplify so that there was enough voltage to turn on the devices.

Despite what problems that came forward this project was able to be completed both with software design and hardware design and therefore a heating and cooling system for a house can be constructed.

Appendix