IMPLEMENTATION OF WIRELESS SENSOR ACTUATOR NETWORK ON A SCALED SUSTAINABLE HVAC SYSTEM

By:

[Nirav Patel (NYP7) & Fisal Sayed (FAS57)]

ECE 4760 Final project

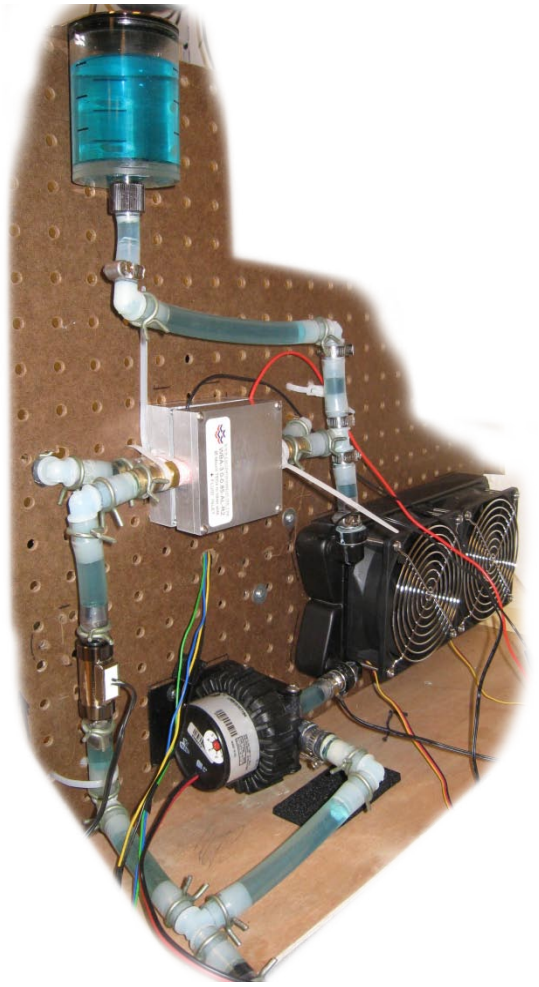


TABLE OF CONTENTS

Acknowledgement

This report covers our work that we carried out towards our final ECE 4760 project. We would like to express our deepest gratitude to Professor Bruce Land and Professor Brandon Hencey for their valuable inputs and comments on the project.

We would also like to take this space to thank our team mates Justin Dobbs for his support with the PI controller code and Jaina Mehta for her help explaining how Zigbee works. We also want to thank our TA Michael Lyons for his valuable inputs about the project.

Introduction

This project creates a wireless sensor actuator network for a scaled Heating, Ventilation & Air Conditioning controller rig that exists in Upson 126 in Professor Brandon Hencey's Lab. This project is meant to facilitate data collection from the HVAC rig to Matlab software on a main PC by taking the full advantages of the wide spreading Xbee modules that operate on Zigbee protocol. This collected data helps to do further data analysis, model calibration and optimization without the need to wire a single component to the computer. This solid wireless network enables users to send commands and change set points and tune PID controllers to facilitate running experiments on the rig. Graphical User Interface GUI has been designed and implemented in Matlab to facilitate interaction with the wireless set-up. Decentralized multi-loop PID controllers have been implemented to control different actuators such as Peltier Heater/Cooler, pumps and fans.

High level design:

Rationale:

Buildings operations account for 40% of the total US energy consumption. This corresponds to about $170 Bn worth of cash. One third of this energy is wasted due to inefficiency in buildings controllers. The average household spends about $2000 a year on energy bills, over half of which goes to heating and cooling. To test new building technologies and to bridge the gap between computer simulation and modeling and the full scale buildings, a small scaled HVAC rig was built. This project is carried out to set baseline PID controllers for the HVAC components as well as designing and implementing a wireless sensor actuator network that facilitates two way communications between computer and the rig. This project also simplifies interaction with the rig and acquiring data by implementing a GUI in Matlab, a well-established statistical software package.

Principle of Operation of the Scaled HVAC System

In order to control the existing HVAC rig, one needs to understand the basic operation of the HVAC. The scaled HVAC rig consists of two main loops; the chiller water loop and the condenser loop. Peltier module has been used to remove heat from the chilled loop and convey it to the condenser loop where heat is removed by passing through a forced air heat exchanger in the condenser loop as shown in figure (1). The chiller water loop consists of a water block attached to the cold side of the Peltier Module. The water flowing through the water block acts as a heat source at the side of the thermoelectric module. The chilled water CHW supplied from the water block goes through a cooling coil available in the conditioned space. This CHW picks up heat in the conditioned space by having a fan blowing air through the cooling coil. The hot water returning from the cooling coil is pumped by a circulating pump to go through the water block (chiller) and this process repeats itself.

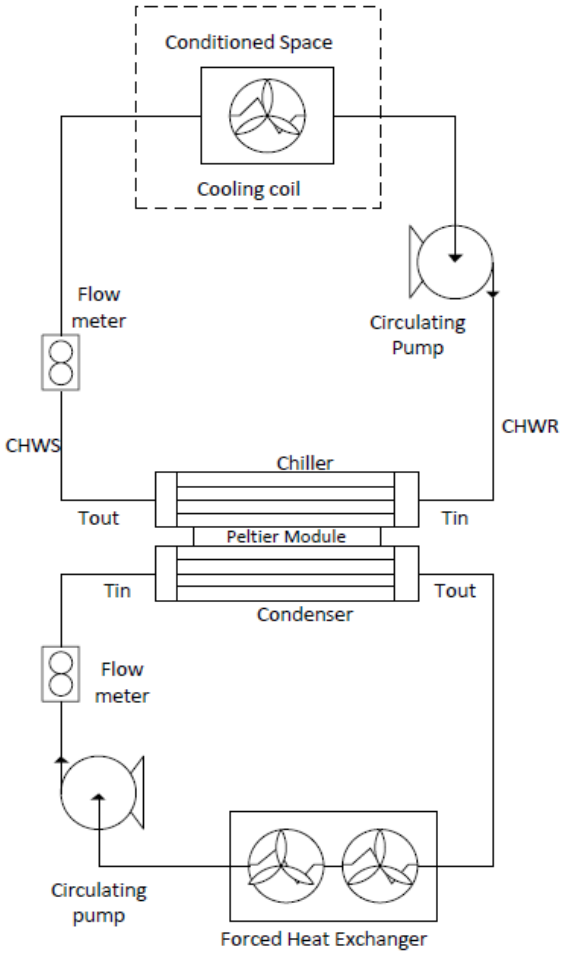


Figure (1) Chiller & Condenser loops of HVAC

The condenser loop consists of water block attached to the hot side of the Peltier module. Water flowing through the water block acts as a heat sink at the hot side of the thermoelectric module. Heat is being removed from the condenser side of the Peltier and being pumped through a double heat exchanger that cools it down using fans. The more heat removed from this loop the better the Peltier performs, the more we can cool down the chilled water loop

PID controller

The most common controller used in the HVAC industry is the proportional, integral, Derivative PID controller. In brief, Proportional, Integral, Derivative PID controller is a feedback controller that helps to attain a set point irrespective of disturbances or any variation in characteristics of the plant of any form. It calculates its output based on the measured error and the three controller gains; proportional gain Kp, integral gain Ki, and derivative gain Kd. The proportional gain simply multiplies the error by a factor Kp. This reacts based on how big the error is. The integral term is a multiplication of the integral gain and the sum of the recent errors. The integral term helps in getting rid of the steady state error and causes the system to catch up with the desired set point. The derivative controller determines the reaction to the rate of which the error has been changing. In most of the HVAC systems, it is not necessary to use the derivative part of the PID, hence in this project, only PI controller has been designed and used. The final output of the controller (U) is calculated using the following equation:

$$U = Kp * e(t) + Ki \int e\, dt + Kd \frac{de}{dt}$$

The signal value U is sent continuously to the driving circuit with every corresponding new output begin measured as the process continues. Table () summarizes the pros and cons of each term of the PID controller.

Controller	Pros	Cons
P	• Easy to Implement	• Long settling time • Steady state error
PD	• Easy to stabilize • Faster response than just P controller	• Can amplify high frequency noise
PI	• No steady state error	• Narrower range of stability

Table (1) : summary of the affect of changing PID functions

Saturation and Anti-windup

The wind up action happens when the output of the controller reaches saturation (255 for 8bit PWM). Due to the error is still positive, the integral part continues to increase causing more inputs applied to the system. This causes the control signal to remain in saturation and the feedback loop is basically broken. This windup action is avoided by restricting the output signal of the controller (U) to be within the acceptable range of the actuator. Here, we have restricted the output signal between 0 to 255 (8bit PWM signal).

PID Controller Tuning Parameters

There are a couple of strategies on how PID can be tuned; this includes trial and error tuning method, Ziegler-Nichols tuning method etc. Our approach was calculating the controller output function U such that the final output signal (U) applied to the actuating circuit was divided by the number of shifts determined in PID structure. This is a systematic way of changing the PID parameters. We started by 7 shifts to the right; meaning we divided the output signal by 2^7 or the value 128. This way we are half way through either increasing division factor by 128 or reducing it by 128. We started by making an initial guess of the value of the Kp gain where Ki=Kd=0. We noticed that when Kp had a large value, we had a faster rise time, but an increase in overshoot. To improve on the effect of Kp, an additional Ki was also set to a small value. Adding integral gain had a great effect on reducing steady state error between the desired temperature and the actual temperature. The following table gave us a rough idea of how changing each PID parameter could change our system response.

Parameter Increase	Rise time	Overshoot	Settling Time	Steady-state error
Kp	↓	↑	Small Change	↓
Ki	↓	↑	↑	Great reduce ↓
Kd	Small Change	↓	↓	Small Change

Table (2) PID controller parameter characteristics on a fan's response

Multi Single input Single Output SISO PID loops

After establishing a functioning single input, single output PI control which was the simplest in design since we used data from one sensor to control one actuator, we moved to designing a single input, multiple output SIMO PI control as shown in figure (2). In this case we used one data from one temperature sensor to control both the Peltier Heater and the fans on the heat exchanger in the condenser loop. In this case, both controllers were aiming to meet one set point, the advantage of such a method is that we could reach to the set-point faster than before but amount of overshoot set point increased. To overcome such a problem, we detuned PI gain parameters and after tuning both controllers, we were able to get a very good response.

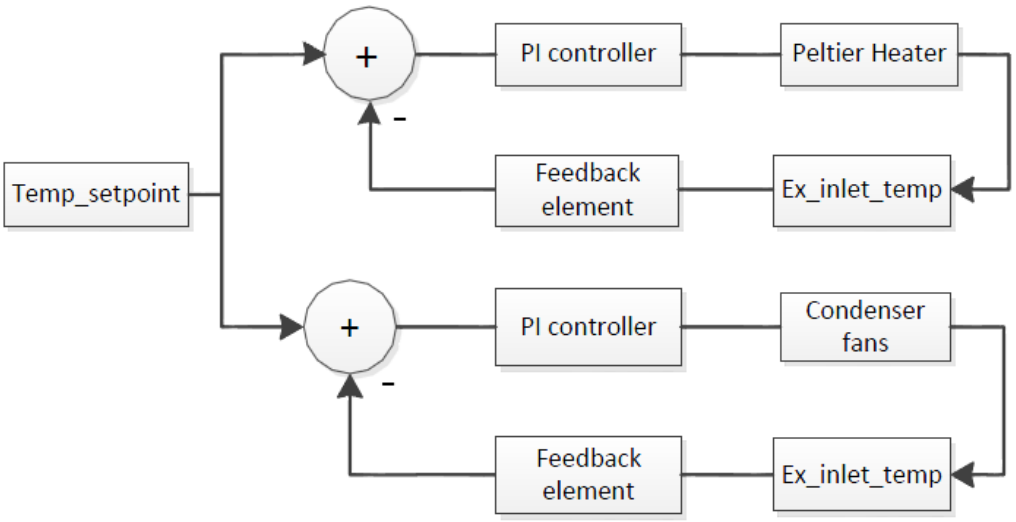


Figure (2) Two SIMO PID loops

Table 3 summarizes the best gains at which we got the best response with less oscillation and faster rise time.

	PID type	Kp	Ki	Max	Min
Peltier controller	PI	20	1	255	10
Fan Controller	PI	20	1	255	0

Table (3) PI controller final gain values

After establishing a working SIMO control, we decided to implement two decoupled PI loops i.e. MIMO control. We worked on controlling the temperature of the chiller water supply loop. The first PI loop was to control the temperature of the Chilled water supply T_CHWS by controlling the Peltier Module. The second PI control loop was to control the temperature of the air supply AS by controlling the chilled side pump. The slower the flow rate, the longer the water stays in the cooling coil, the hotter the CHWR return temperature gets, This CHWSR acts as a load on the Peltier cold side, resulting in conveying more heat to the hot side of the Peltier. Figure (3) shows MIMO PI loops where one loop acts as disturbance. Such a system is difficult to tune specially, when one controller acts as a disturbance to the other controller.

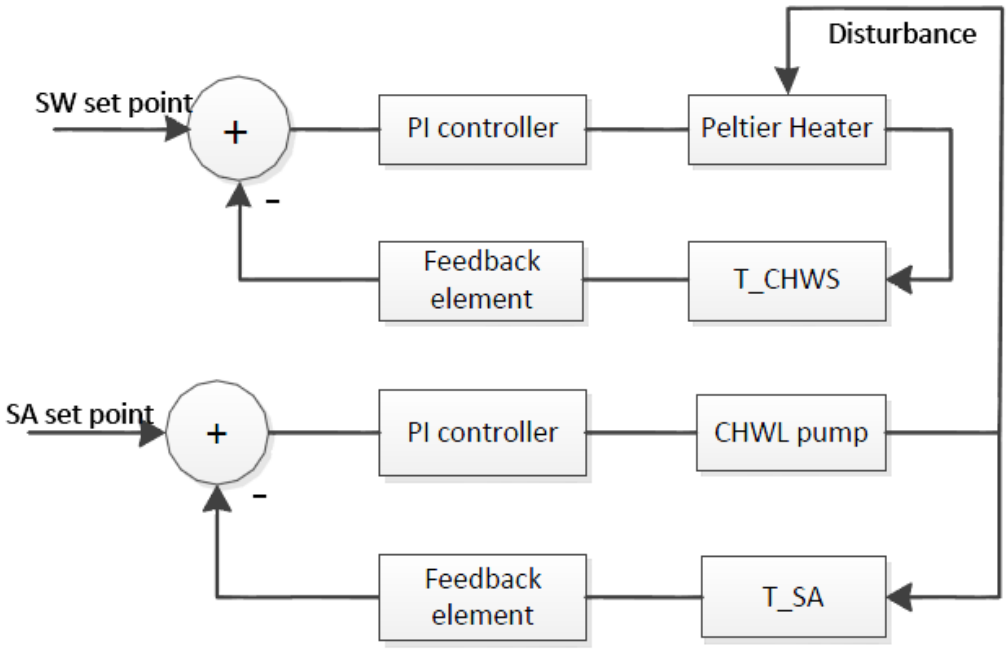


Figure (3) Two MIMO PID loops

Communication standards/protocols

UART

The UART stands for Universal Asynchronous Receive/Transmit. UART is used in reference to the standard protocols like RS 232, Rs 422 or RS 485. The most common RS-232 protocol has been defined to operate at a high voltage level relative to the supply to the controller. This was in conjunction to the industry requirements so that it is immune to noise. Hence it needs a separate driver circuitry which translates the low voltage output of the UART of the controller to a recognized level for RS-232 communications.

Wireless ZigBee Protocol:

For the wireless communication, we used the Zigbee Protocol. This protocol is used with low power wireless radio in communication systems which require low data rate, high battery life and secure networking. The Xbee module manufactured by Digikey implements this protocol. We have been using the Xbee modules which are directly compatible with the Arduino boards and work on a standard UART interface. Hence communication with the PC at the receiving end is also very straightforward. Also sending data from the controller is also as simple as putting data on the UART transmitter line of the controller. The module takes care of the implementation of the protocol and transmits the data. All the modules having the same Personal Area Network ID can form a mesh network. Hence the user does not need to actually care about connecting several nodes to a master node. This is the biggest advantage of using this module.

The Zibgee protocol is somewhat new in the communication Arena, compared to the RS 232. This protocol was designed for digital radios which worked at low power and low data rates that can be used for Wireless Personal Area Networks (WPAN). It has a data rate of 250 kbps, probably sufficient for sensors designed for intermittent data transmission. The operating frequency is 915 MHz in the US. The implementation of this protocol is cheaper than some of its counterparts for WPAN like Bluetooth. The advantage of using Zigbee is that it can automatically create an ad-hoc network amongst the nodes. Hence the user does not have to worry about setting up a network; just setting the correct parameters in the

module is all that it takes for the node discovery and network formation. The typical protocol frames for some modes we have used/tested are discussed in the appendix.

First version: API version

The Zigbee modules work in different modes. API mode is one of them. This mode sets the module to accept and put out data on the UART in a predefined frame which is basically defined by the protocol. This gives more encapsulation to the actual data and is really useful in setting up a mesh network wherein we have to deal with data from multiple nodes. The details of this protocol are defined in the appendix. The drawback to this protocol is that the decoding of data becomes complex at the receiving end. But the advantage is that each frame has a payload which consists of information on the senders address, number of data bytes in the frame and checksum for error checks. Hence there is a tradeoff between the complexity of the design and the features offered by the design.

Second Version: Transparent version

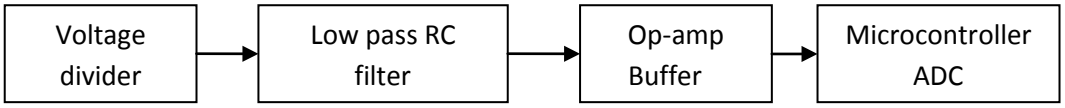
The transparent version, as the name suggests is more of a user friendly form of communications. In this mode, it just puts out the data it received from another node on to the UART. Vice versa, it accepts the data from the serial terminal without having to form any framing operations manually. This makes the decoding tasks at the receiver really simple and straightforward. Nevertheless in this mode, the Zigbees' communicate within them using the appropriate protocol, just that now its hidden from the user. The drawback here is that the module truncates all the frame information which is provided in the API mode, and instead just puts out the actual data transmitted on the serial. This reduces the complexity but hides the sender information from us. Hence to implement a mesh network, we may need to send out the address of the sending node before transmitting the actual data.

We tried using both the API and the transparent (AT) mode. This project is a research project which needs to be followed up for a few semesters. We had to keep in mind the changes which will be made to the system over the coming semesters by different students. Hence using the transparent mode will help them make these changes a bit smoothly compared to the API mode. So we propose using the AT mode keeping in mind the future developments necessary in the system.

Hardware circuit design:

Temperature sensor circuit:

Thermistors have used as a low cost temperature sensor which is constructed of solid semiconductor material which exhibit a positive or negative temperature coefficient. We have used a 10k NTC Thermistors. Since Thermistors are basically thermal resistive devices, a signal conditioning circuit is necessary to convert the change in resistance into a change in voltage. All Thermistor sensors have been connected in series with a 4k7Ω. The output of the voltage divider is then filtered and isolated using a buffer which has a very high input impedance and low output impedance to isolate the voltage divider circuit from the microcontroller. MCP6004-I/P Quad Op Amp was chosen because it has four op-amps on one chip. This made it easier to build and made the circuit layout smaller. A look up table in the microcontroller was used to convert each temperature. Temperature sensors had to be calibrated before being used.



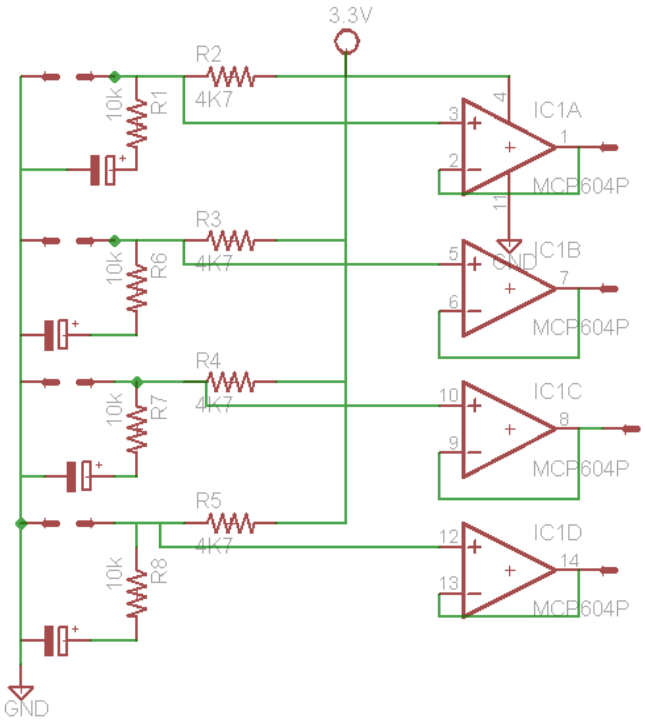


Figure (4) signal conditioning circuit

Actuator driving circuit

In this project, there were six actuators to be controlled; three fans, 2 pumps and Peltier Heater/cooler. All of these actuators have motors in them and require a high current to be driven. A low side gate driver IXDN604PI has been used to drive a power MOSFET. This gate driver has two outputs can sink 4A of current which producing voltage rise and fall times of less than 10ns. The input to these drivers is virtually immune and provides sort of protection to the microcontroller. To slow down the fast switching of the gate driver, a 10Ω resistor is connected at the gate of the driver. When signal is applied to the gate driver, it turns on the MOSFET which in turn pulls the load down to ground. The input channels of the gate driver have to be pulled down by a resistor to keep it from floating around. MOSFET (IRLB3034) has been used to drive the actuators; this MOSFET was chosen because it can sustain up to 195A through the drain and it's $R_{DS\ (on)}$ =1.4mΩ; dissipating about 1Watt when I_D is 28A (the Imax required by the Peltier). Two MOSFETs have been connected in parallel to provide the required current of the Peltier heater/cooler

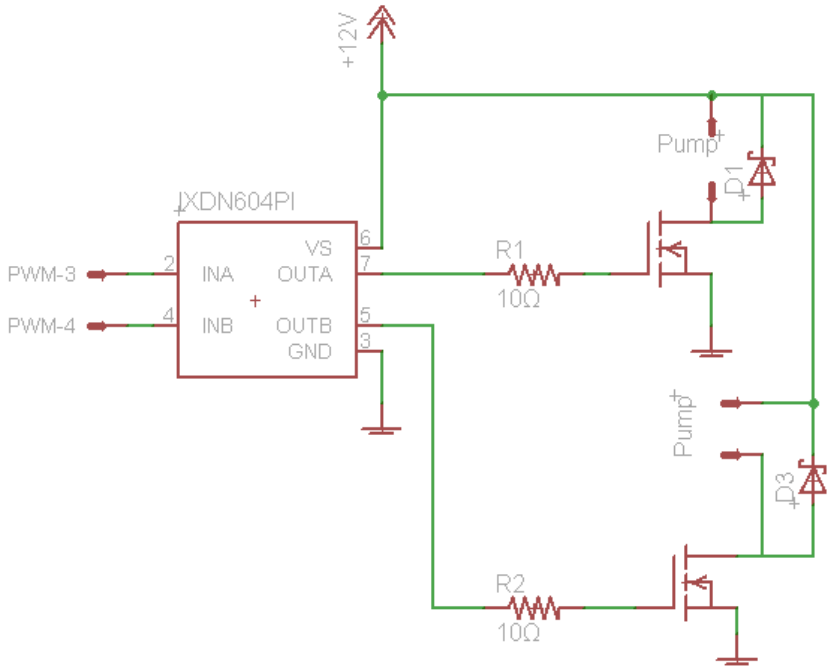


Figure (5) Actuator Driving Circuit

Microcontroller board

We have used the Arduino Fio board for satisfying the Microcontroller requirement as shown in figure (6). It houses a Mega328 controller. Various features offered by the board, especially the interface with the Xbee modules for wireless communication is a very advantageous feature for us. Also it has all essential peripherals like the ADC channels, PWM channels and UART lines drawn out from the board. Hence it acts as a tiny module with all the features essential to our project. Although this board provides a good hardware platform, it comes with a bootloader program which enables us to use a very simple programming language. This language is meant for beginners and is very user friendly, but is highly inefficient. We have been using C to program the board but on several occasions it created a compatibility issue and caused the system to perform randomly. This was a trade-off where we had to combine the use of the Arduino language and C in order to maintain compatibility and efficiency. Also, since it is a development board, we do not have access to all the port pins on the controller. Hence in cases where we need to use the pins which are not directly available on the board, it can fail to meet our requirements. Also the crystal on the board is configured as 8 MHz, which can is well below the maximum limit on clock frequency. So it means that we are not utilizing the full capabilities of the controller.

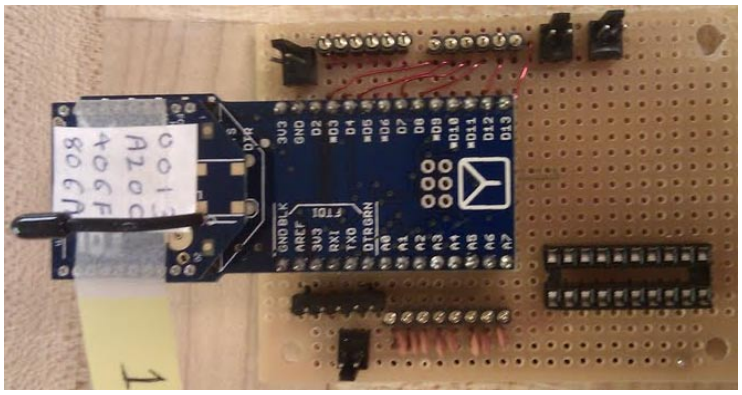


Figure (6) Controller board

Peltier Module Power Measurement

Measuring the power consumed by the Peltier Heater/Cooler is a feature that we decided to add to the hardware design of the system. The maximum current the Peltier module requires at 12V is 28A, with this in mind, we decided to use ASC 712 current sensor which has a current sensing range of ±30A and it is based on the Hall Effect Current Sensing. Figure (7) shows the schematic diagram of IC connection. This IC requires 5V supply. We had to add a linear voltage regulator to provide the necessary voltage. The output of sensor had to be stepped down to a range of 0-3.3V using a voltage divider circuit. To accurately measure the power consumed by the Peltier module, we had to take instantaneous measurements of current and voltage. Voltage measurements were taken exactly at the terminals of the Peltier module to take into account any voltage drop across the Peltier wires. Both voltage and current were sampled only when the Peltier is on and to be able to do that, we had to synchronize the ADC measurement in the microcontroller such that it starts sampling when the PWM signal is on as shown in figure (8)

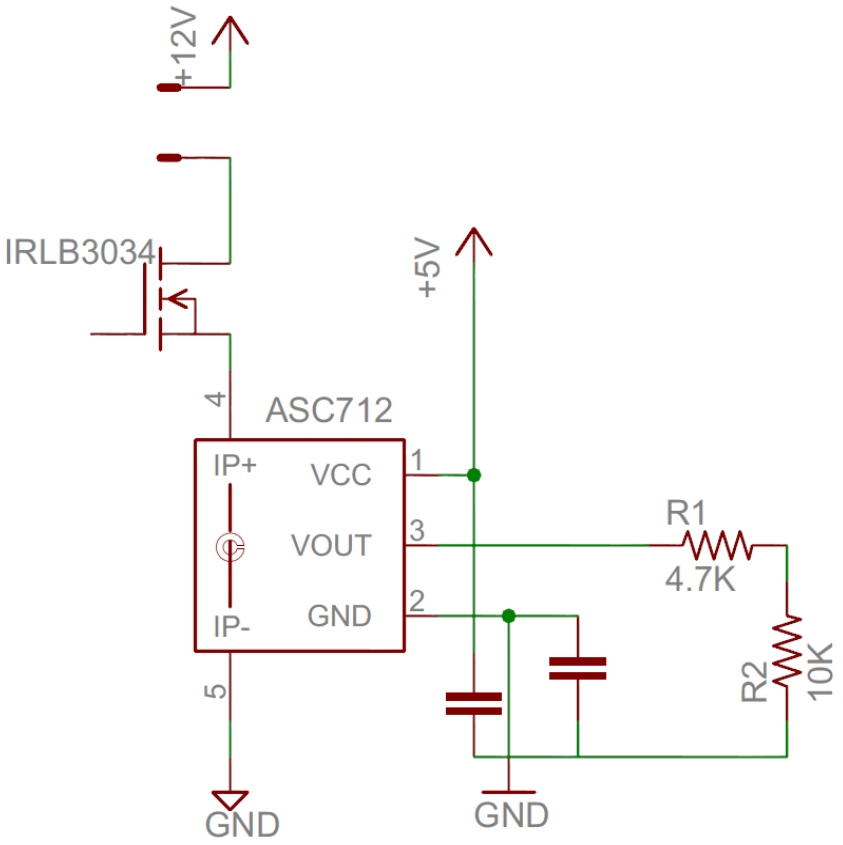


Figure (7) current sensor circuit

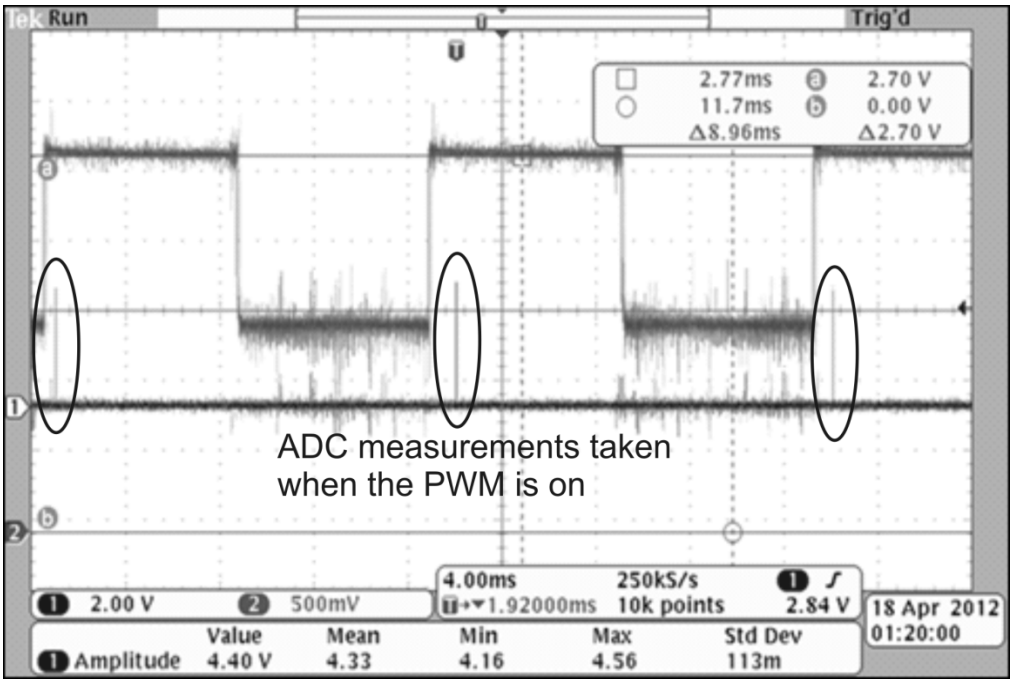


Figure (8): PWM & ADC synchronization

Hardware Testing
Stage 1

This project involved a lot of hardware testing and debugging since the circuit has 4 different voltage levels; these are 3.3V, 5V, 12V and 24V. These voltage levels existed to meet the voltage supply needs by different components in the circuit. To get 24V needed by the Peltier Heater, we had to connect 12 fat power supplies in series. These two power supplies introduced a considerable amount of noise to the system. Having different voltage levels on one circuit board was a big challenge in getting the circuit to work. After we finished the design of the layout of individual circuits, we started thinking about the complete layout of the circuit. The layout of the circuit was done such that different high power components were separated from the low voltage part of the circuit. All Actuators were connected to the circuit board from one side using Molex connectors. Depending on current required by each actuator, 2, 4 and 6 pin connectors were used. Soldering the circuit was done on different stages to make sure that each part of the circuit worked fine. Each actuator output was tested by connecting a resistive load to it. This was done to make sure that there was no short circuit existed and that the gate driver and the MOSFET function as expected. In addition, we did not want to blow out any of the actuators until the circuit is completely tested.

Temperature sensors signal conditioning circuit and the controller board were each soldered on a separate piece of board that was mounted on the main actuators board. This was done to meet the modularity requirement of the project and to facilitate the replacement of the controller.

Even though these circuits were meant as prototyping and only as a mean to get the HVAC rig functioning, we paid our best efforts to avoid any nasty connections. Temperature sensors circuit conditioning circuit was soldered using a Kester 331 flux solder which is free from any toxic metal and it is a water soluble flux core solder which is used with a flux bottle which worked well for soldering up the board without rosin residue. This flux gave a much cleaner result because it completely washes away under the tap. The only downside is that we had to dry the circuit board with compressed air before we can test it.

Stage2

The second stage of the hardware testing involved connecting actual loads to the circuit. These loads are 3 fans, 2 centrifugal pumps and 1 peltier Heater/cooler module. Most of these loads are inductive and switching inductive loads generate transient voltages and spikes of many times the steady state value. These transients can destroy the controller especially in our case, they share the same ground. scotchy diodes were connected across both pumps. These diodes were reverse biased such that they will allow current to pass through them when the pumps are turned off.

Since we had a noisy power supply, a highly voltage rated electrolytic capacitor was connected across the supply to filter out some of the noise. A think ground plane was made especially for the Peltier heater since it passes 28A through the ground plane and that causes ground bouncing. Figure (9) shows a ground bouncing at the controller terminal which was caused by the high power switching. This causes wrong readings, and can cause the controller to fail.

Having a fat low inductance ground plane helps tremendously. We connected 0.1uF decoupling capacitors for every IC between Vcc and ground to bypass the supply inductance. Also, since the Op Amp signals from the signal conditioning circuit travel a

distance to be connected to the microcontroller ADC channels, a 10Ω resistor is connected at the Op Amp output so that the capacitance of the cable does not load the Op Amp and cause it to oscillate.

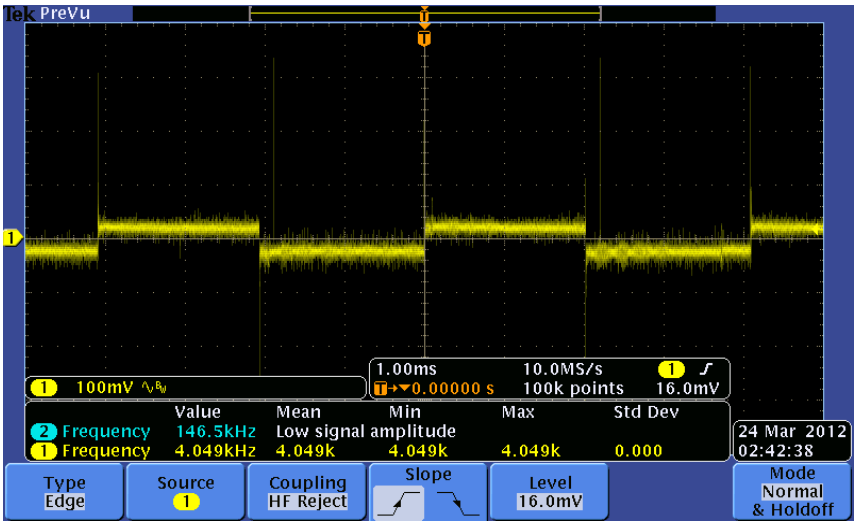


Figure (9) Ground bouncing

Software Design & Testing

In the software part of this project, we first started by setting up the microcontroller to measure temperature readings. To accomplish this, we had to configure the Analog to Digital Converter ADC. We started by reading one ADC channel and afterwards, we scanned all ADC channels. We were able to convert resistance change of thermistor to temperature using an equation that we got from the curve fitting the resistance change verse temperature plot. This was inefficient as it involved a lot of floating point calculation which takes longer time to calculate and overloads the microcontroller; instead we established fixed point arithmetic where we used a look up table that looks up a corresponding temperature based on the ADC 10 bit value. After being able to measure temperature, we then configured timer0 to generate a time interrupt of 1ms. This helped us in calculating

the flow rate of each loop based on the frequency measurement we got from the flow sensors. Flow sensors were connected to digital pins D12, D13, so each time there is a change in the status of any of the pins, we incremented a variable which after 1000 times interrupts, we convert these pulses into frequency. Having both temperature and flow rate measured facilitated us introducing a single input single output PI controller for the chiller loop pump to control the follow speed in that loop. We were also able to run a single temperature PI controller on the chiller loop to control the temperature of the air supply.

Stage1: This stage was pretty straight forward as we integrated the configuration of ADC, timer interrupt, and PI loop together. In addition, we started tuning the temperature PI controller to give the best response with minimum overshooting. Setting the PI parameters were a compromise of how fast we wanted to get to the set point verses oscillation around the set point. During this stage, we have also implemented safety checks such that if the Peltier Heater temperature exceeds 90°C, then we switch it off as well as if there is no flow in the condenser loop; we switch the pump off to protect it against running dry. The following flow chart indicates the high level design of the software.

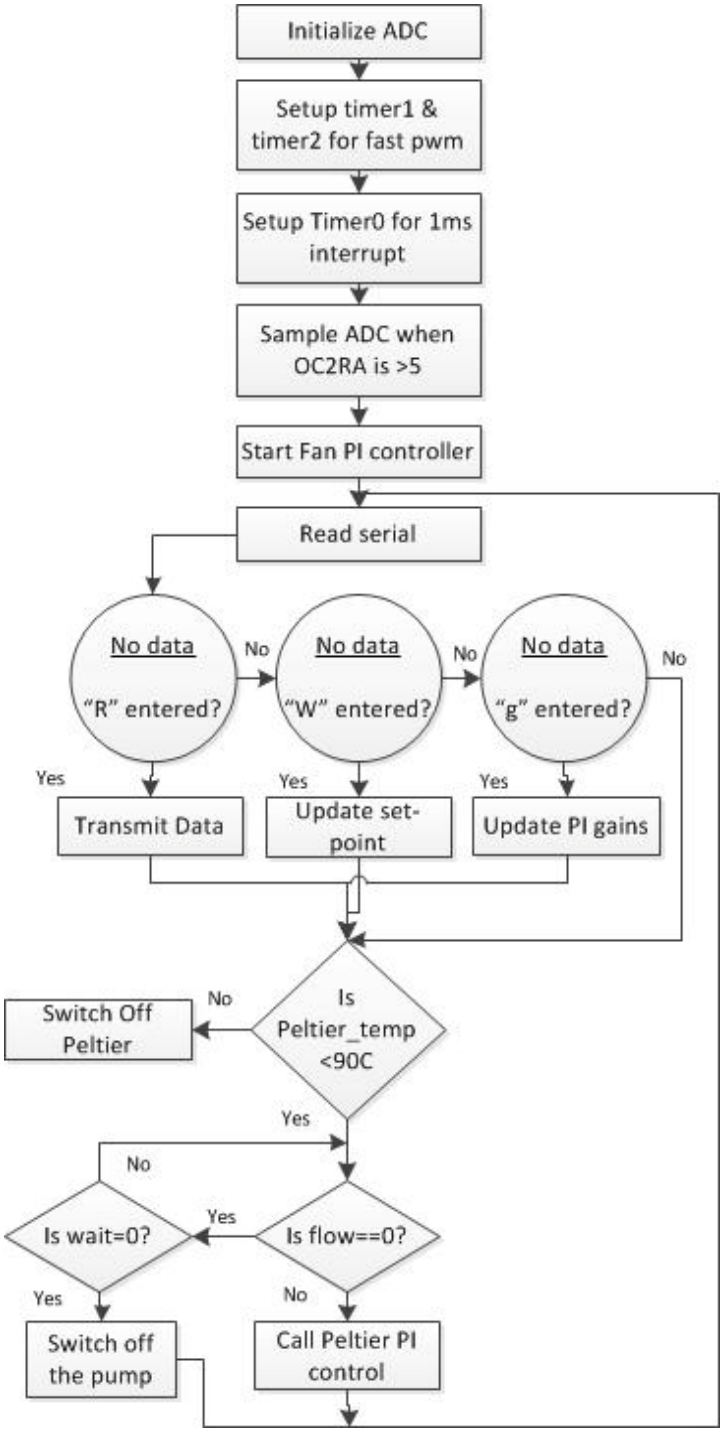


Figure (10) Controller flow chart

Stage 2: In this stage we started the implementation of wireless communication between the PC and the controller. To implement this feature, we later on designed a GUI in Matlab and also an interrupt driven UART for the controller as well. In the first stage of our design, we needed to reprogram the controller to observe the response of the system under various settings. This also introduced a risk of destroying the controller from the physical or electrical damage (static charge that our bodies hold). The implementation of wireless communication helped us overcome this difficulty. In this stage we utilized a serial terminal program, TeraTerm to transmit and receive data wirelessly using Zigbees. Though it was helpful, we did not have any control over incoming data. Nevertheless it was the first step towards designing the GUI.

Stage 3: To overcome the limitations of stage 2 design, we designed a GUI in Matlab. In the primary design, we implemented a control over the amount of data we receive. We did this by restricting the Matlab code to a certain number of user entered reads. It was not optimal compared to controlling the data flow from the controller itself, but it proved to be a good tool for debugging the performance of the system. In the later part of this design stage, we added a feature to transmit the user entered setpoint to the system.

Stage 4: Achieving total control over the data transmitted and received was the goal of this part of the design stage. All these features enhanced the control any user had over the system.

The features we added in this stage are:

- Number of reads and interval between reads request sent to the controller
- Introducing acknowledgement signals from the receiver to confirm a successful data transfer
- Implementation of failure checks and display of appropriate error messages.

Final GUI Design

Improvements: GUI

To make the wireless interactions between the controller and the computer more user friendly, we decided to design a GUI which implements the functions. Initially we had functions which needed to be called from the command line in MATLAB which was a tedious task for a user unknown to the system. Hence we plan to have a GUI designed which implements the commands to acquire data wirelessly and also command the controller to reach a desired set point on the system.

The GUI was designed using the GUIDE – a layout editor available in Matlab. The key features of the design are pointed out in figure (10) and explained briefly.

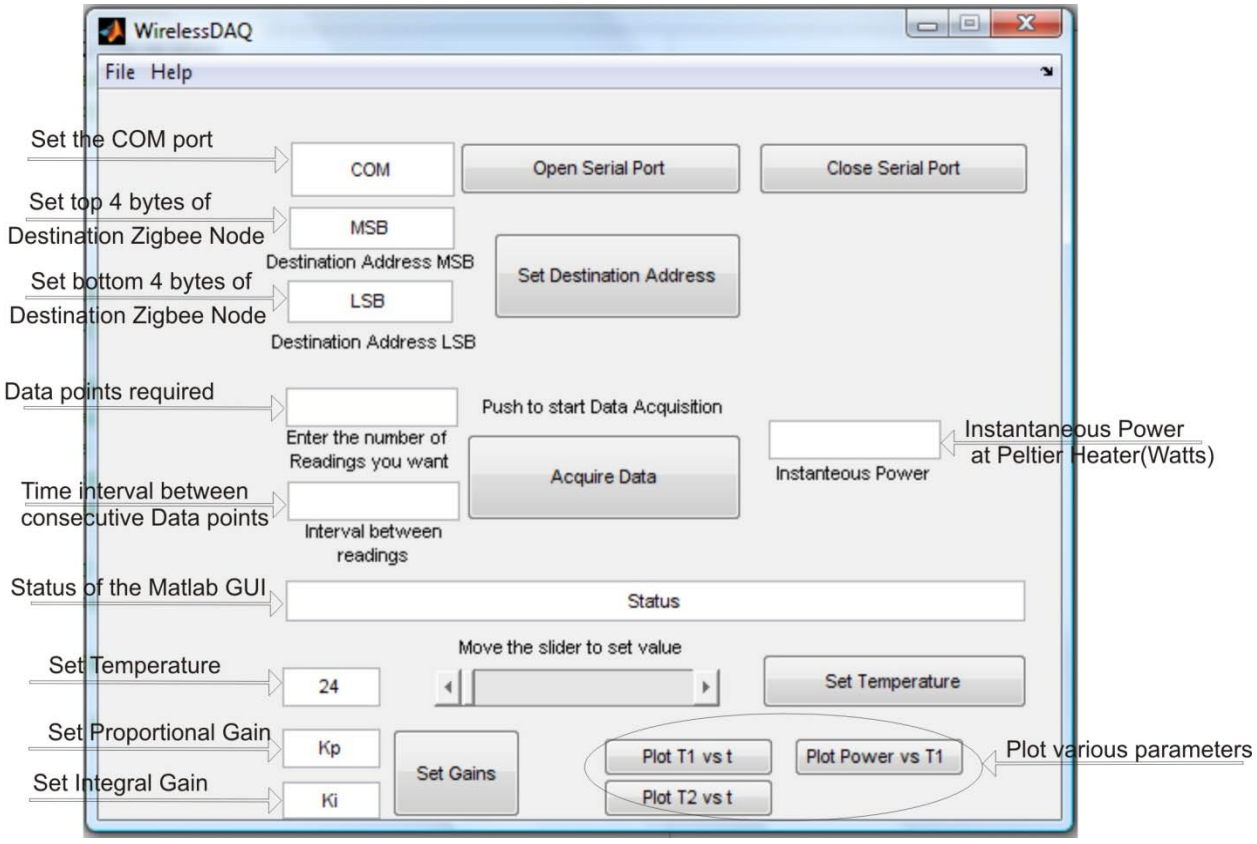


Figure (10) : GUI design layout

Setting the COM port of the connected Zigbee Module: As the serial port is now obsolete in most personal computers and laptops, the use of USB-Serial converters is increasing. These devices on the USB emulate a serial port and hence do not have an assigned COM port address (eg COM1, COM2 and so on). Hence on different computers we may have different port addresses. To make the GUI more user friendly, we included an option where in the user can enter the comport value and then open the serial communication using the 'Open Serial Port' button.

Setting the Destination Address: The Zigbee module can hold a single destination address. In a mesh network where we have multiple nodes and want to communicate with all of them, we can change the destination address on the Zigbee and hence communicate with the desired node.

Setting the number of data points required: To implement a controlled communication, we added this feature of supplying the number of data points required. This feature commands the controller to send the data only when asked by the user and also in the quantity asked by the user.

Setting the time interval between reads: Since we are dealing with controlling the temperature here, sending out data every couple of milliseconds just accumulates redundant data. Hence to control the interval between successively transmitted data point, we provide the user with the feature to enter this value.

Setting the temperature: This feature sets the desired temperature for the PI control loops. It has been implemented using the slider bar and the text box. So a user can update the value at any of these objects. The range of temperature that has been provided is from 24˚C to 37˚C in steps of 0.1˚C.

Setting the gains: Changing the gains and seeing the response of the system in essential in choosing the best combination of gain values. Before we had this feature, we needed to reprogram the controller to change the gains and see the effect of this change. This was a very tedious task and involved removing the controller from the board and inserting it

again. Hence we decided to add this facility which enables any user to change the gains and observe the effect of this change.

Plotting the data after it has been received: Once the data has been received and saved to a mat file, we have the feature of plotting the data against time. The two plot functions we have plot the two temperatures against the time based on the user defined iterations and intervals between the iterations.

Power: Based on the readings of the current and voltage sensor, we calibrated the ADC to resemble the instantaneous power consumed by the system. After extracting the data, we calculate the power consumed at that instant and display it on the GUI window.

Protocol Used for communication: Since we are using the Zigbee Module in the transparent mode, the format data received is identical to the one received in the standard serial communications. Hence to distinguish the different data points from each other we have a start byte (126 = 0x7E) and an end byte (126 = 0x7E). Hence each data point is encapsulated within the pair of start and end bytes. A typical data frame would look something like (0x7E)(Data)(0x7E). Data is sent out by the microcontroller in a preprogrammed sequence and hence can be decoded and sorted out at the receiving end.

We use an interrupt driven UART at the controller which can decode the commands sent by the user from the GUI.

Decoding within the Matlab GUI:

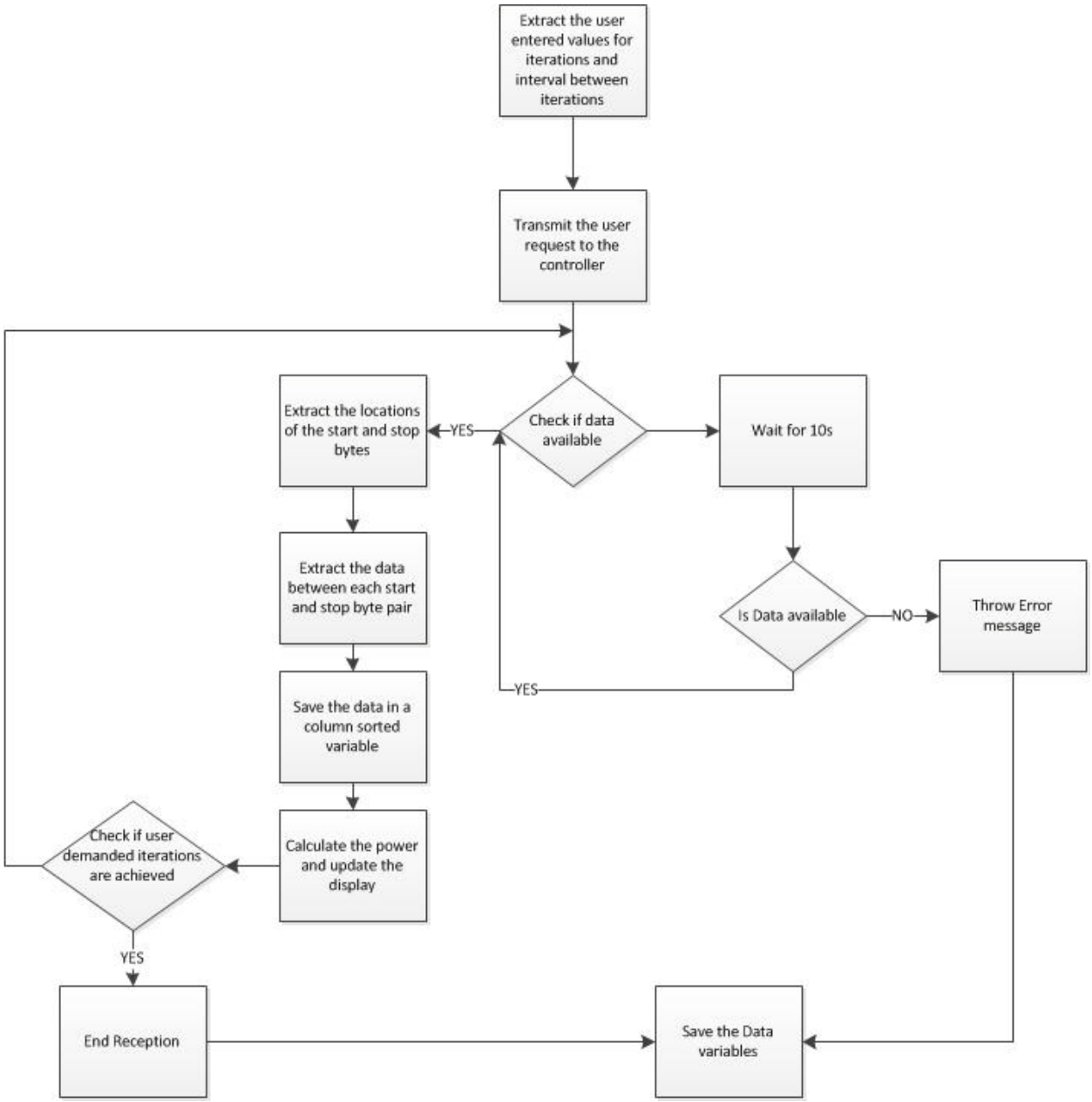


Figure (11): MATLAB GUI DECODING

Decode Logic at Controller:

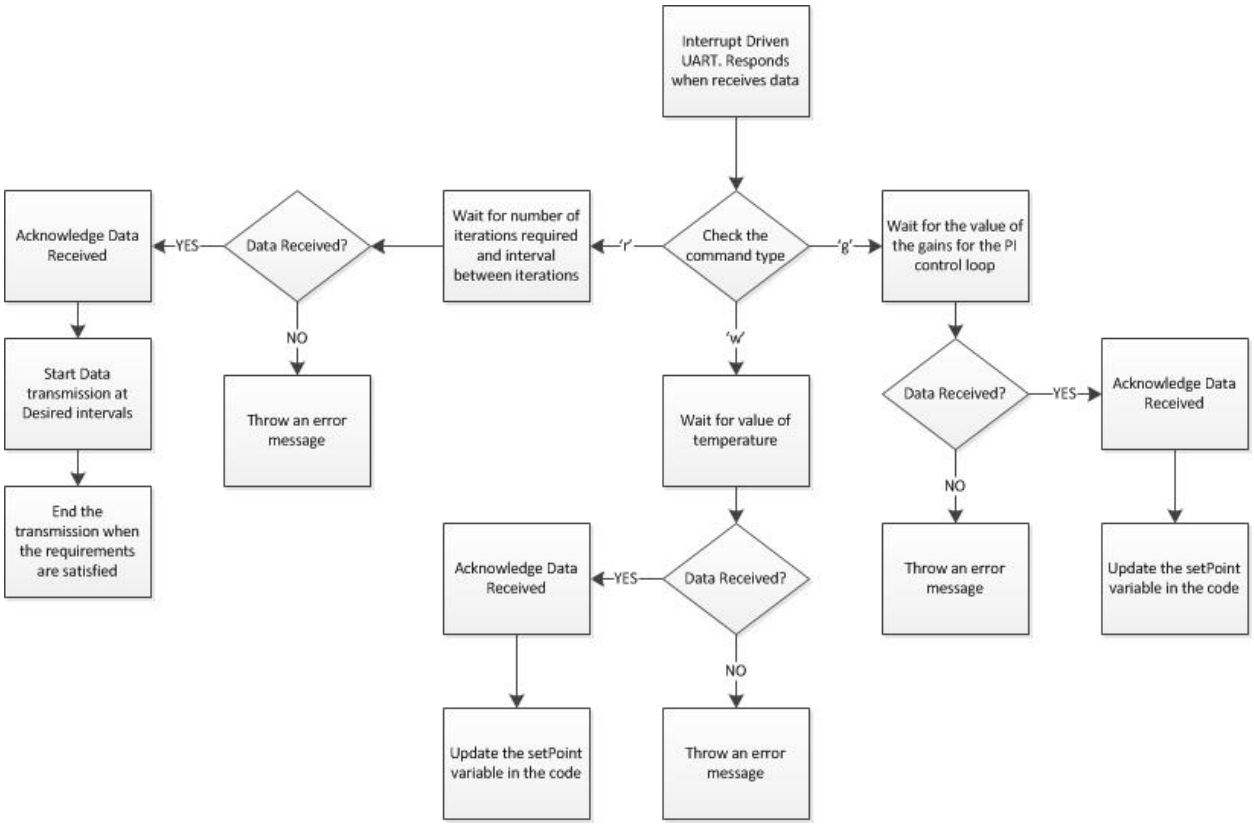


Figure (12): Logic decoding in Arduino Controller

Design Results

The results obtained from this project were a completely functioning scaled HVAC system that can be controlled wirelessly using Zigbee protocol. A wirelessly tunable decentralized multiple input, multiple output PI controllers were successfully implemented and tested. The details of the results obtained are as follows:

- Temperature sensor calibration

To accurately measure the temperature of the HVAC setup, all Thermistor sensors were calibrated and compared to their datasheets. We calibrated the sensors by immersing each sensor in to a cup that has boiling water in it and as water cools down, we take measurements of the temperature of the water using an accurate thermocouple temperature sensor and we measure the corresponding resistance of the Thermistor at that temperature. Due to the fact that we had many temperature sensors in the system, we only used one look up table for all of them.

- Peltier Power Measurement

To accurate measure the power consumed by the Peltier Heater, we synchronized the measurements of the Peltier's voltage and current such that we only sample the ADC when the PWM (OC2RA) goes high. In addition, we measured the voltage of the Peltier exactly at its input terminals to avoid any voltage drops across its wires. This gave us a fairly accurate instantaneous power measurement of this power hungry component.

- Proportional, Integral Controller Response

The response of the PI controllers were very satisfactory since we were able to obtain the set point with in ±0.7°C. Figure (13b) shows the response of implementing one PI controller on the Peltier heater. We were able to get a better response when two PI loops were implemented as shown in Figure (13a). Using GUI interface greatly helped us tuning the PI controllers wirelessly.

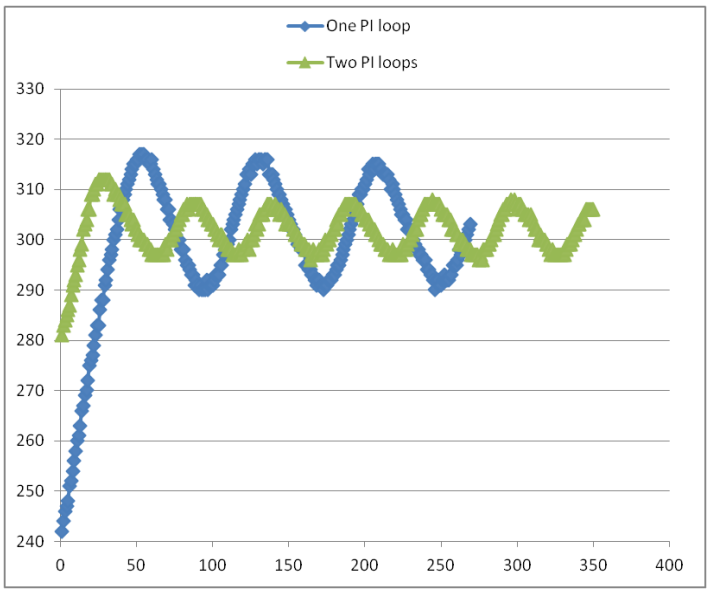


Figure (13a): Two PI loops response Vs one PI

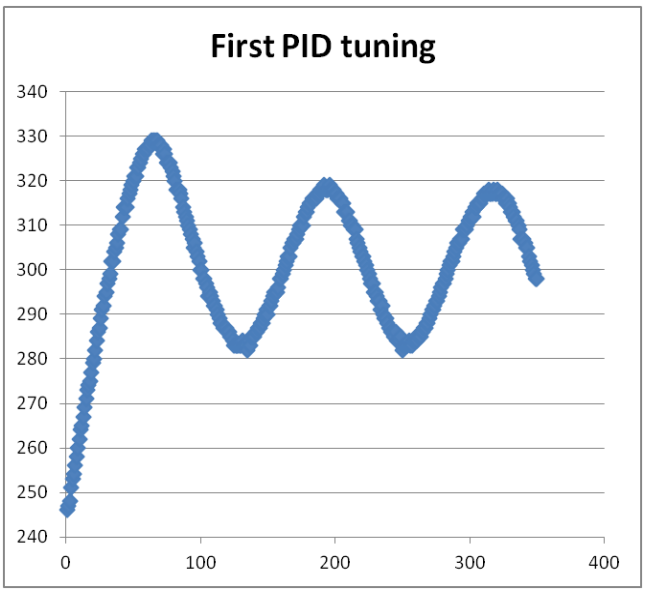


Figure (13b): Peltier PI controller response

Measured data from both the condenser loop pump and the chiller loop pump were collected to be compared and contrasted with the pump models that have been designed using Modelica language. Further analysis will be carried on this project to calibrate HVAC models and build a Model Predictive Controller that will take into account systems dynamics to optimize best set point to be sent to the local PI controllers.

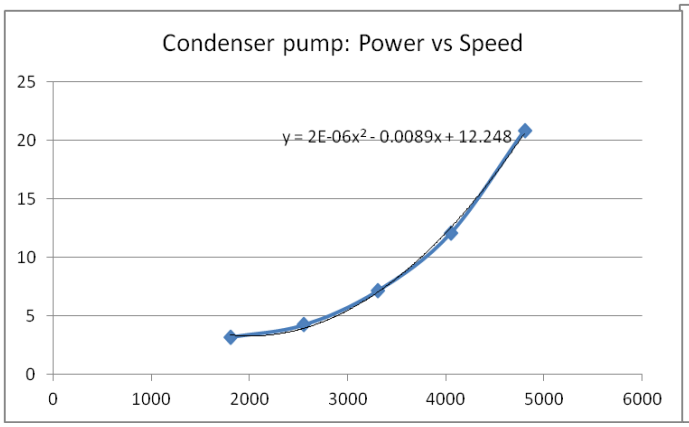


Figure (14a) Power vs speed of MCP655

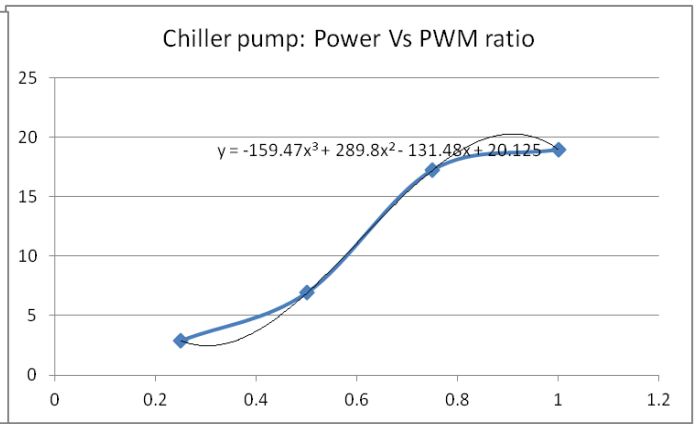


Figure (14b)Power vs PWM ratio of x35

Figure (15a) Power consumption of Peltier Module

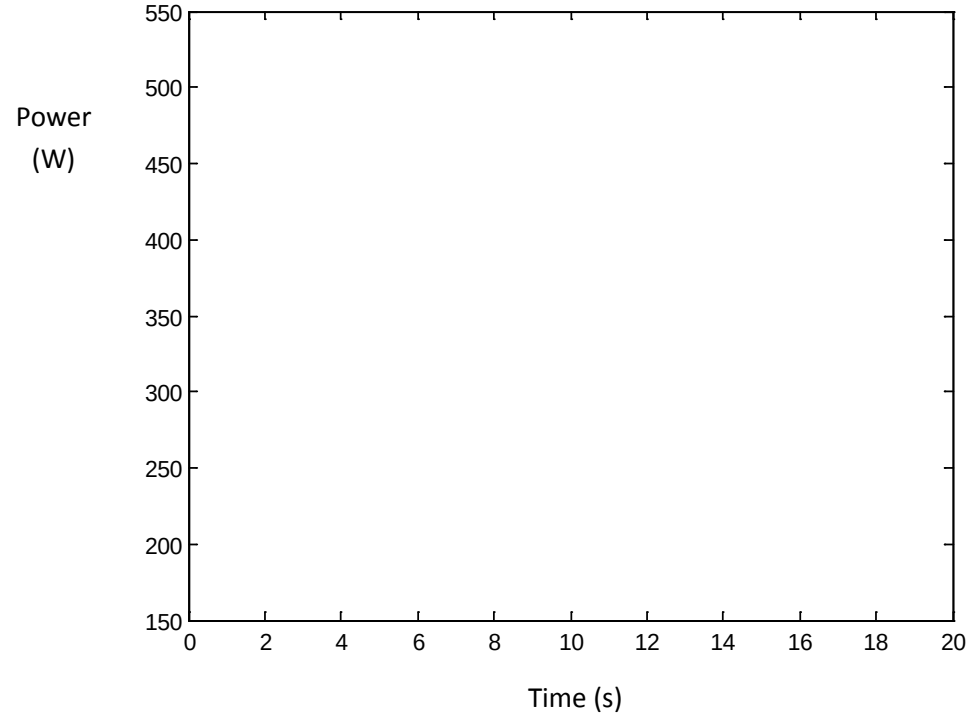
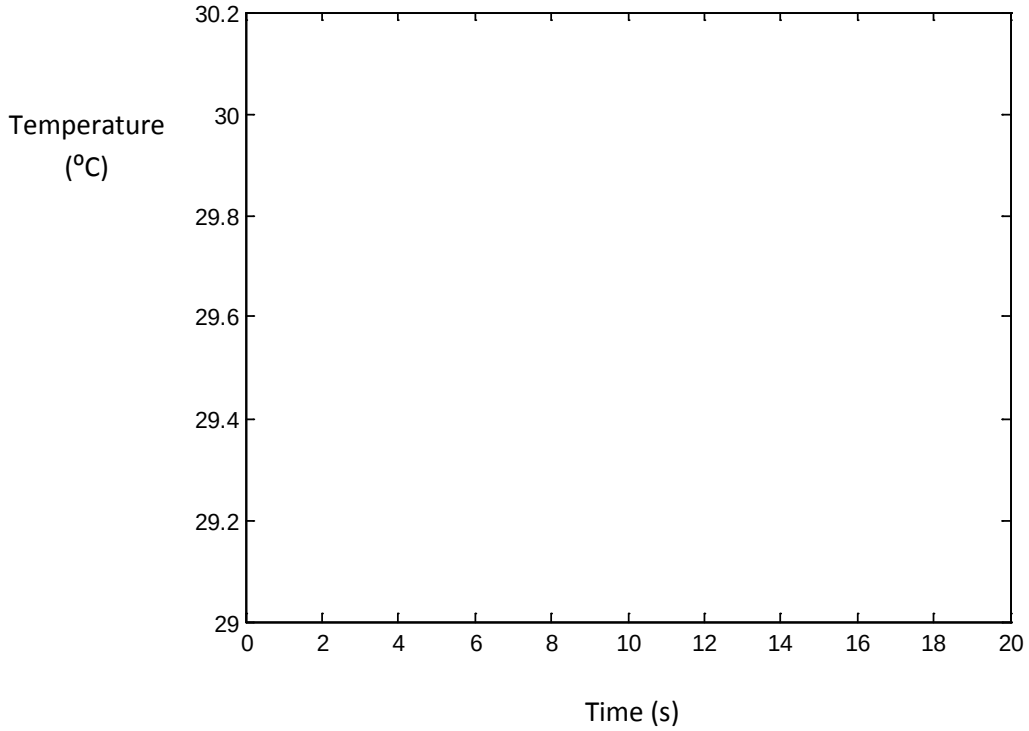


Figure (15b) Temperature Transient response of Peltier



Safety

This project involves a high DC power application. Hence it is very important to keep the safety concerns in mind while interacting with the system.

Peltier Heater, Pump and the Heat exchanger fans in the hot loop: This component is sensitive to the current. Hence the duty cycle of the PWM applied defines the average power being relayed to the device. Whenever the control for the Peltier heater is revised in the controller code, it is a safe practice to check the output of the controller pin controlling the device on the oscilloscope by connecting a simple resistive load. This gives an idea about the performance of the system and helps us determine how the actual device will react to such an input. Even a minor programming error can lead to overheating the heater and destroy the peripherals as well.

Not only malfunctions related to the Peltier can cause it to overheat. The performance and control over the peripherals can also cause the system to go haywire. For instance in the pump in the hot loop is not working, the peltier will not be able to transfer the heat to the heat exchanger and hence heat up very quickly. Same is the case with the failure in the control of the heat exchanger fans. So as a standard procedure it is good to check the control signals on the oscilloscope before hooking up the devices.

Solution we have:

- We check for the temperature at the Peltier heater. If it exceeds the limit of 90˚C, we turn off the peltier. It cannot restart until reset manually. This enables the user of the system to know that there is some problem in the system.
- The second check is whether the pump in the loop is working or not. If it is working without any fluids, it may burn out the coil of the pump. Hence we check the flow in the loop and if it is zero we turn off the pump and since the pump is off, the peltier will be heated and so we need to turn that off as well.

  As the previous check, this is a manual reset operation to alert the user about any issues with the system.

Current Sensor IC ACS 712: This is a surface mount component designed to carry 30 A current. But the connectors on which it rests may not be as capable as the IC. So there is a need to cool down the IC to avoid burn out.

Solution we have: Currently we have the cooling fans within the power source we are using, directed towards the circuit board. This basically cools down everything there in and works as a temporary solution.

Multi-value Voltage supplies for various components: On this board, we have components working at 3.3 V, 5 V, 12 V and 24 V. Hence it is very important to take care not to short out two different voltage values while trying to use oscilloscope, multimeter or something as simple as hook up wires.

Solution we have:

- We have designed the connectors in such a way that the user can connect them only in one possible way. Hence there are no chances of shorting the Vcc and ground terminals.
- Avoiding the shorting of different voltage sources is up to the skills and care taken by the user.

Conclusions

The results of this project were very satisfactory. We established a very responsive HVAC system that is wirelessly controllable. We established a very good understanding of multivariable feedback controllers design and tuning. There were some issues with hardware debugging and ground bouncing and this gave us very rich hands on experience especially when we are pulse modulating high power components and inductive loads. We were able to identify some hardware limitations which are listed in the future improvements of the system. The implementation of the wireless network gave us a concrete understanding of different protocols and the drawbacks of each one.

Important Design Considerations & possible design improvements

Grounding:

When dealing with switched mode operation on devices, it is necessary to have a sufficiently large ground plane to suppress the transients. Especially in high power circuits like the one we have, needs a large area to ground the current flowing in the circuit. If the ground plane is not sufficiently large, frequent switching causes the ground level to bounce and hence affect the performance of the microcontroller. There is also a possibility of unpredictable behavior of the controller or its peripherals.

Optical Isolation:

As discussed above, grounding can be an issue on the performance of the low power devices (controller, sensors etc) when using along with high power switching devices (pumps, heaters etc). Hence it would be a good idea to isolate the low power and high power devices to stabilize the performance of both. An opto-isolator (capable of accommodating the desired bandwidth of the switching signal) would serve this purpose by transferring the low power switching signal optically at the high power end. This would practically isolate the two power levels. IC's are readily available for this purpose. Current design does not have this feature but it would be good to have it during future improvements.

Possible Improvements

Even though we did our best in building the hardware, it is still considered as a prototype and may not be very robust if a lot of students will use it. The current connectors between the controller and its inputs, output is not robust. All connector wires and jumpers should be replaced with a soldered wires or use ribbon cables to get a solid connection. The circuit needs to be contained in a transparent box to protect it from water leakage in the system and a fan is required to cool down the current IC sensor and the Peltier power connector. We are currently using one look up table to get the temperature readings for all thermistor sensors, instead one could have included a look up table for each sensor, however execution time and memory space needs to be considered. Finally, to protect the actuators from any sudden current transients, one could use fuses in line with each actuator to protect it from surge currents.

Appendix A: controller commented code

Appendix B: Matlab GUI commented code

Appendix C: cost details

No.	Part	Unit Cost	Quantity	URL
1	Arduino Boards	$25.00	3	http://www.sparkfun.com/products/10116
2	Xbee Wireless Modems	$22.00	3	http://www.sparkfun.com/products/8665
3	INS- FM18 Flow meter sensor	$30	2	http://www.koolance.com/water-cooling/product_info.php?product_id=1170
4	Peltier Heater/Cooler	$56.50	2	http://www.shop.customthermoelectric.com/searchquick-submit.sc;jsessionid=49BADFC26E6C47EDBF65E2A8D246654D.qscstrfrnt01?keywords=400
5	Swiftech MCP655 circulating Pump	$76.00	1	http://www.sidewindercomputers.com/swmc12vdcpu.html
6	Swiftech MCP-35X pump	$110.00	1	http://www.sidewindercomputers.com/swmc12vdcpu1.html
7	LCD display screen	$13	1	http://www.sparkfun.com/products/255
8	Power Supply for Peltier Cooler 24 V	$210.00	2	http://www.trcelectronics.com/Meanwell/se-1000-12.shtml
9	$2	$2.00	4	http://www.datasheetcatalog.com/datasheets_pdf/M/C/P/6/MCP6004.shtml
10	lm2937et Current regulator	$1.00	1	http://www.datasheetcatalog.com/datasheets_pdf/L/M/2/9/LM2937ET-12.shtml
11	Water Blocks	$79	2	http://www.customthermoelectric.com/Water_blocks.html
12	IC Gate Driver non-inverting 8Dip	$2	2	IXDN604PI
13	MOSFET	$4	5	IRLB3034PBF
14	3.3 voltage regulator	$1		www.digikey.com
15	7805 voltage regulator	$1		www.digikey.com
16	12V votlage regulator.	$1		www.digikey.com
17	ASC current sensor	$4		www.digikey.com
18	Condenser radiator	$70	1	http://www.koolance.com/water-cooling/product_info.php?product_id=813
19	Chiller loop exchanger	$39	1	http://www.frozencpu.com/products/5323/ex-rad-106/Black_Ice_GT_Stealth_120_Radiator_-_Blue.html?tl=g30c95

Appendix D: Pin References:

Connected Actuator/Sensor	Pin on the Arduino Board	Pin on the Controller
Peltier Heater	D11	PORTB.3
Pump in the hot loop	D4	PORTD.4
Heat Exchanger Fans (hot loop)	D9	PORTB.1
Pump in the cold loop	Any of the available pins	
Fan in the cold loop	D3	PORTD.3
Peltier Outlet Temperature	A3	PORTA.3
Heat Exchanger (hot loop) outlet Temperature	A4	PORTA.4
Peltier Hot Side Temperature	A1	PORTA.1
Fan Out temperature (cold loop)	A0	PORTA.0
Current Sensor	A6	PORTA.6
Voltage Sensor	A7	PORTA.7
Flow Sensor (hot loop)	D12	PORTB.4
Flow Sensor (cold loop)	D13	PORTB.5

Appendix E:
Troubleshooting and Understanding the GUI:

COMPORT – Enter the address of the Comport after looking it up in the device manager.
Possible Errors:
1. Invalid ComPort. Enter a valid Comport
If no COMPORT value is entered or a non-existing COMPORT is entered, you may get this error message.
Look up the correct name of the port from the 'Device Manager' under properties of 'Computer'. Try to open the Serial Port after the change.

Destination Address – Enter the address of the wireless node you want the data from.
Possible Errors:
1. 'Enter 8 Hex Characters of the MSB':
If you enter less or more than 8 characters, this error will appear in a message box. Revise the address and enter it again in the edit box.

Number of Reads and Read Interval – The user enters the number of data point he wants and the time interval between consecutive data points.
Possible Errors:
1. Entering non integer values:
Entering floating point values here would cause an error message to pop up. Rectify the entered number.
2. Entering zero:
Entering Zero will give an error message and would default the number of reads to 20.
3. Entering alphabets in the edit box:
If the input is alphanumeric or alphabetic, an error message will pop up. Please rectify the input and retry.

Gains Kp and Ki – The user enter the gains for the PI loop controlling the Peltier heater.
Possible Errors:

1. Floating point of alphanumeric or alphabetic input by the user:

If such values are entered it will cause the GUI to throw an error message pop-up. Rectify the values based on the instruction and retry.

Default Settings: The default settings have been configured for the following:

1. Number of iterations is 20 if invalid value is entered and used
2. The read interval is set to 1000 ms if invalid value is entered and used.
3. Temperature is set to 24˚C.

Before using the Set Gain Function in the GUI, one must take care to enter valid values. If valid values are not used, it may lead to unpredictable behavior of the system

References:

Books/ Datasheets:

- Robert Faludi "Building Wireless Sensor Networks" 1st edition, December 2010.
- Oludayo John Oguntoyinbo "PID CONTROL OF BRUSHLESS DC MOTOR AND ROBOT TRAJECTORY PLANNING AND SIMULATION WITH MATLAB/SIMULINK" , 2009
- Atemga 328p datasheet: http://www.atmel.com/Images/8271S.pdf
- Datasheets for ACS712, lm 7805, LD33V, IRLB3034, IXDN604PI

Background sites:

- www.Avrfreaks.net
- www.arduino.cc
- Zigbee general information: http ://en.wikipedia.org/wiki/ZigBee
- Matlab GUI using GUIDE: http://www.mathworks.com/help/techdoc/creating_guis/f8-998197.html