HVAC Control and Feedback System 2.0

Group 2

Steve Jones

Mathew Arcuri

Elroy Ashtian, Jr.

Jerthwin L. Prospere

Spring – Summer 2011

Due: 08-05-2011



**Table of Contents**

1 Introduction 8

1.1 Executive Summary 8

1.2 Motivation 9

1.3 Objective 10

1.3.1 The Remote Sensing Unit 10

1.3.2 PIC Microcontroller 11

1.3.3 LCD Display and ARM processor 11

1.4 Specifications 12

1.4.1 Detailed Design Features 13

1.5 Division of Labor 14

1.5.1 Coding 16

1.5.2 Milestones 17

1.5.2.1 Senior Design I 17

1.5.2.2 Senior Design II 18

1.6 Financing 19

2 Research 19

2.1 Research Methods 20

2.2 High Level Control Unit 20

2.2.1 ARM Microprocessor Based Master Main Controller 22

2.2.1.1 Processor Compatibility with Slave and other components 23

2.2.1.2 Constraints on size of Microcontroller 24

2.2.1.3 Analog to Digital Conversion and Input/Output ports 24

2.2.1.4 Communication Protocols / Integrated Peripherals 25

2.2.1.5 Memory Type 25

2.2.1.6 Cost 25

2.2.1.7 Voltage and current requirements 26

2.2.2 High Level Control Unit 26

2.2.2.1 Gumstix Overo 26

2.2.2.2 Linksprite 28

2.3 Microcontroller as Slave 32

2.3.1 PIC 33

2.3.1.1 Microcontroller Compatibility with other Components 34

2.3.1.2 Analog to digital and digital to analog converters I/O 34

2.3.1.3 Memory Type 35

2.3.1.4 Development tools 35

2.3.1.5 Overall architecture of system and Cost 36

2.3.1.6 Voltage and current requirements 36

2.3.2 ATMEL 41

2.4 Secondary Microcontroller/ Remote Sensor: 42

2.4.1 General Description 42

2.4.1.1 Operating Range 43

2.4.1.2 CPU Characteristics 43

2.4.1.3 I/O Characteristics 44

2.4.1.4 Cost of the original unit 44

2.4.2 Xbee vs Zigbee Wireless 46

2.4.3 Wired Connection 46

2.5 Wireless and Internet Connectivity 47

2.5.1 Overview – Wireless and Internet Connectivity 47

2.5.2 TCP/IP 47

2.5.3 Operating System - Wireless and Device Driver Implementation Maturity 48

2.5.4 Server 50

2.5.5 Web Application 51

2.5.6 Client Side Application 53

2.5.7 RSS Weather Feeds 53

2.6 Power System 55

2.6.1 Main Unit 55

2.6.1.1 Rectifier 56

2.6.1.2 Distribution 56

2.6.1.3 Vent Servos and Mood Scents 57

2.6.2 LCD Screen 57

2.6.3 Remote Sensor Unit 57

2.6.4 Dampers 60

2.7 Sensors 61

2.7.1 Temperature/Humidity 61

2.7.2 CO2 Sensor Applications 62

2.7.2.1 Senseair® CO2 EngineTM K22-OC 64

2.7.2.2 Senseair® CO2 EngineTM K30 66

2.7.2.3 Senseair® CO2 EngineTM K-33 ELG/BLG 67

2.7.2.4 Cost 69

2.8 Seer Ratings 69

2.9 Serial Connections 71

2.9.1 UART 71

2.9.2 I2C 73

2.9.3 SPI 74

3 Design 75

3.1 Software Design 75

3.1.2 Embedded Software 80

3.1.2.1 Components and Functionality 81

3.1.2.2 Microchip and ARM 83

3.1.2.3 ARM Processors 84

3.1.2.4 Microchip PIC microcontroller 85

3.2 Wireless 88

3.2.1 RF Communication with Secondary Unit 88

3.2.1 WIFI internet connectivity 89

3.3 Sensors 89

3.3.1 Temperature/Humidity 89

3.4 LCD Screen 94

3.4.1 Environmental Conditions for Xenarc LCD Monitor 95

3.4.2 Electrical Characteristics 96

3.5 ARM Microprocessor Board 96

3.5.1 Electrical Characteristics 97

3.6 Power System 98

3.6.1 Main Unit 98

3.7.2 Secondary Unit 100

3.7 Relays 100

3.8 PCB 103

4 Prototype 104

4.1 Vendors 105

4.1.1 Xenarc Technologies Corporation 106

4.1.2 Microchip 106

4.1.3 Sensirion 106

4.1.4 DigiKey 107

4.1.5 Senseair 107

4.1.6 CO2Sensors.com 107

4.2 Schematics 108

4.2.1 Main Master Control Unit 108

The entire schematic is large and almost incapable of reading in its entirety, which is why the above sections were partitioned. Below in Figure 49 is the entire schematic with the added relays. 111

4.2.2. Secondary Unit Schematics 111

4.3 Main Control Unit 112

4.4 Remote Sensing Unit 114

4.5 Relays 116

4.6.1 High Level Control Unit and Main Control Unit 117

4.6.2 Remote Sensing Unit 117

5 Testing 119

5.1 Explorer 16 Development Board 120

5.1.1 Communicating with ARM 122

5.2 Safety Considerations 122

5.3 Equipment 122

5.4 Environment 124

5.5 Relays Testing 125

5.6 Relays Testing 125

5.7 Final Specifications and Requirements 125

5.7.1 Final Specifications 129

5.7.2 Final Requirements 131

6 Administrative 133

6.1 Milestones discussion 133

6.2 Budget Discussions 134

Table of Figures

[**Figure 1 Product Diagram** 14](#_Toc300258420)

[**Figure 2 Division of Labor** 16](#_Toc300258421)

[**Figure 3 Milestone Sr. Des. 1** 18](#_Toc300258422)

[**Figure 4 Milestone Sr. Des 2** 19](#_Toc300258423)

[**Figure 5 Overo Air Com (Reprinting Permission requested from Gumstix)** 27](#_Toc300258424)

[**Figure 6 LS6410 ARM11 Android Dev Kit** 28](#_Toc300258425)

[**Figure 7 Beagleboad** 29](#_Toc300258426)

[**Figure 8 Pandaboard (permission granted under the Creative Commons License)** 32](#_Toc300258427)

[**Figure 9 Minimum Connections** 40](#_Toc300258428)

[**Figure 10 Encapsulation of data in TCP/IP** 47](#_Toc300258429)

[**Figure 11 Wireless Network Selection in Ubuntu** 49](#_Toc300258430)

[**Figure 12 Wireless Security Configuration in Ubuntu** 50](#_Toc300258431)

[**Figure 13 RSS feed example code** 54](#_Toc300258432)

[**Figure 14 K30 Schematic Highlighting UART** 58](#_Toc300258433)

[**Figure 15 Sensair K33 Schematic highlighting I2C connections.** 59](#_Toc300258434)

[**Figure 16 I2C internal connections on the K30/K33**  Note: There is an addition 10R resistor between output and the DVCC pin that is not shown (reprinted with permission from Senseair) 60](#_Toc300258435)

[**Figure Figure 17 Senseair® CO2 EngineTM K22-OC Module** 64](#_Toc300258436)

[**Figure 18 Senseair® CO2 EngineTM K22-OC Module** 65](#_Toc300258437)

[**Figure 19 Senseair® CO2 K22-OC: Depicting alarm status of CO2** 66](#_Toc300258438)

[**Figure 20 Senseair® CO2 EngineTM K30 Module** 67](#_Toc300258439)

[**Figure 21 Sensair® K-33 BLG powered via battery** 68](#_Toc300258440)

[**Figure 22 Short Cycle Energy Loss** 70](#_Toc300258441)

[**Figure 23 UART Transmission Character Frame** 71](#_Toc300258442)

[**Figure 24 UART Transmission with CPU** 72](#_Toc300258443)

[**Figure 25 I2C Data Transmission Timing Diagram** 74](#_Toc300258444)

[**Figure 26 Data Transmission from Slave to Master** 75](#_Toc300258445)

[**Figure 27 Use Case diagram for HLCU** 76](#_Toc300258446)

[**Figure 28 Use Case diagram for HLCU** 77](#_Toc300258447)

[**Figure 29 Proposed design for the HLCU User interface Main Tab.** 78](#_Toc300258448)

[**Figure 30 Use Case diagram for MCU** 79](#_Toc300258449)

[**Figure 31 Use Case Diagram for the wireless sensor** 80](#_Toc300258450)

[**Figure 32 Basic Functionality** 81](#_Toc300258451)

[**Figure 33 Hierarchy** 84](#_Toc300258452)

[**Figure 34 State Diagram for Mobile Web Site** 87](#_Toc300258453)

[**Figure 35 Main Layout** 88](file:///F:\SeniorDesignpaper-revised-final.docx#_Toc300258454)

[**Figure 36 Schematic Design of PIC24 with CO2 UART CONNECTION** Note: WiFi connections removed – denoted by parenthesis. 92](#_Toc300258455)

[**Figure 37 HLCU Communication Setup** 98](#_Toc300258456)

[**Figure 38 24VAC Full Wave Rectifier** 99](#_Toc300258457)

[**Figure 39 Rectifier Output** 99](#_Toc300258458)

[**Figure 40 Relay open** 101](#_Toc300258459)

[**Figure 41 Relay Closed** 102](#_Toc300258460)

[**Figure 42 Relay Overview** 103](#_Toc300258461)

[**Figure 43 Secondary PCB** 103](#_Toc300258462)

[**Figure 44 Main PCB** 104](#_Toc300258463)

[**Figure 45 Prototype Unit** 105](#_Toc300258464)

[**Figure 46 Main Controller Schematic** 109](#_Toc300258465)

[**Figure 47 Power Schematic** 109](#_Toc300258466)

[**Figure 48 Sensor Connections** 110](#_Toc300258467)

[**Figure 49 Main Schematic** 111](#_Toc300258468)

[**Figure 50 Secondary Schematic** 112](#_Toc300258469)

[**Figure 51 High Level Explorer 16** 121](#_Toc300258470)

Table of Tables

**Table 1 Code Responsibility** 17

**Table 2 Beagleboard Revisions** 30

**Table 3 PIC pin layout** 40

**Table 4 Secondary Unit Cost** 45

**Table 5 Resistor values of the K20/K21/K22/K30** 93

**Table 6 Important descriptors of the Xenarc LCD panel** 94

**Table 7 Specifications for 7 inch Xenarc LCD Panel** 95

**Table 8 Environmental conditions** 95

**Table 9 Xenarc Specs** 96

**Table 10 HLCU features** 97

**Table 11 DC Power Specs** 98

**Table 12 Accessory Voltage** 113

**Table 13 Operating Voltage Range for Components of Remote Sensing Unit** 115

**Table 14 Current use** 115

**Table 15 Parts List** 129

**Table 16 System Control Specs** 130

**Table 17 Final Microcontroller Specs** 131

**Table 18 HVAC Components and the Corresponding ASHRAE Naming and Color Standards** 132

# 1 Introduction

## 1.1 Executive Summary

Cost effective energy conservation continues to be a strong motivator of many innovations today. Electrical Engineers are in a unique position to directly help this. The original HVAC control system set out to tackle this problem with a unique handle on the central heating and air of a house. More importantly, this system has many applications beyond residential, leading to commercial buildings. Positively affecting the cost of controlling the climate of an environment can prove to be one of the most significant benefits to overall power use.

Instead of blindly turning on your air conditioner, the system is now smart. It will first run a check, depending on the user settings of the outside temperature versus the inside temperature and may be able to adjust the house temperature in a much more efficient manner. An outdoor sensor was needed, and powered by an external power source sending information wireless by radio frequency. If the user set the system to Max Savings, the first of two air conditioners, the smaller unit, would first try to cool the house, followed by the bigger unit, then the two.

Unfortunately, user input was severely limited in the original unit, not providing as much freedom in behavior as needed. Further, connection to the internet was needed to provide the user the ability to constantly modify the settings desired.

Safety was also a new feature to be installed. An internal CO2 sensor was applied to the main unit, constantly monitoring the internal atmosphere. If carbon dioxide increases too much, not only is an alarm possible, but the entire ventilation of the house. Further addition to cost effectiveness is the air flow. Many people change their air filter in given time intervals. Often this is forgotten and done too late, or done too slow. If air flow/air pollutant sensor was able to more accurately determine necessary interval of filter exchange, not only is money saved, but the air itself is safer. On top of all the cost saving and safety aspects, we are looking to add a setting for the users enjoyment. Mood Scents were planned but not executed earlier. The intent is to be able to control a room’s scent via the main control unit with a dispenser inside of the vent in selected rooms. The user will be able to select the desired times that this will the dispersed.

Lastly, control of all vents in the house will be added. In order to even more control the cost effectiveness of the HVAC unit, select rooms will have their vents opened or closed in selected times. There are already vents with efficient close hatches; it will be up to our project to get the signal to all vents controlling the actual selected temperature of individual rooms. In Max Savings, it could be selected that only one main room needs air control, while some rooms may never be entered. Motion sensors will also be explored.

## 1.2 Motivation

Few things are more aggravating that the heater getting kicked on when the house is not cold because your roommate or spouse has poor blood circulation. Running and exercise can increase blood circulation but the easiest thing for many to do is simply turn on the heat. Likewise it is both the time consuming process of opening windows when it is cold enough to cool your living area with outside air. The other disadvantages of letting fresh air in from windows can be both bugs and safety concerns in that some may feel they might not want to fall asleep or leave the house while the windows are open for several hours. The cost of heating or cooling an entire house when there are more efficient ways of doing so but because those ways aren’t easy or aren’t convenient or for reasons simply because one individual who is in another room sleeping likes it colder or hotter than the other individual in the same house doesn't make fiscal sense. If it were possible to grab an already efficient HVAC system and fine tune it to localize heating/cooling and make it easy and convenient to program the HVAC system for the user efficiency and comfort would go up.

With the surging cost of oil and thus energy more efficient ways of consuming electricity becomes more important and systems that use to be too expensive to consider are quickly becoming attractive investments. Central heating and air account for one of the largest percentages of the energy budget.

Another interest was safety. CO2 accumulation above a certain level can be bad for someone’s health. Having the HVAC system of a house automatically vent the house when harmful gas accumulation occurs is a great idea for preventing harm. We discuss such an idea in the CO2 section of this paper with fires and cooling the house to prevent a worsening situation, however the idea is applicable to other areas. For instance upon air quality detection indicate a need for filtration a system could be designed to have UV lights and filtration via HEPA filters or ventilation of the room only when harmful gases are detected not only make the air cleaner and the environment safer but are more economical than have these systems on at all times.

Going beyond safety and cost efficiency, having the added pleasures in life are a strong motivation. We can paint our rooms different colors, have different TV's playing different shows, and listen to different music. Why not allow our nose to enjoy differences per room? It would be neat to have mood or scent parameters chosen by a user possibly in a per room basis, either activated by the HVAC fan or timed. We would like to take a previous HVAC system, debug it, and further the use and efficiency of the system. Adding web interface through WI-FI was a main goal as it is allowing user adjustable controls. CO2 sensors needed to be added to the house for safety, and the scent feature needed to be implemented. The operating system on the unit needed to go beyond simply display without functionality, but actual functionality and user programmability.

## 1.3 Objective

The system is expected to get data, weather wireless or directly connected, and report to the main control unit. This information will be then transmitted to the ARM microcontroller which will display the data. The ARM microcontroller does the decision making and handles the user interface. The PIC microcontroller interacts with the external devices such as the temperature/humidity sensors and CO2 sensors. These are the elements that were required to satisfy our sponsors, and ourselves:

* The system must do all that the original HVAC 1.
* The system must have internet connectivity.
* The system must have a CO2 sensor to monitor the inside air quality.
* The system must have the addition of Mood Scents. Our sponsor wanted to see the signal coming from the control unit accurately.
* The system should be able to also send signals to open and close vents.
* The system needed to have a completely functional user interface and operating system.
* The possibility of future update of the system software or firmware needed to be inspected and implemented.
* The possibility of adding wired connectivity to the outdoor unit needed to be explored as well as other ways to power unit.
* All bugs and shorts in the original unit needed to be fixed.

### 

### 1.3.1 The Remote Sensing Unit

The remote sensing unit was to be set up outside the building being used. It needed to accurately measure the outside temperature and humidity and transfer this data back to the main control unit. This transfer of data can either be through the original XBee interface or through an added wired connectivity so that buildings with thick walls can still communicate. The reason of getting this data remains the same as in the HVAC series 1, so the system would better be able to figure the most economically efficient way to adjust the inside climate. Unfortunately the original unit had several shorts in the circuit board, requiring several manual modifications. Also, the ability to turn off the unit was solely controlled by a jumper. This needed to be fixed to a switch/button interface.

The old unit was easy to install, and that is a must for this as well. Wired connectivity is not necessary, as it is an exhaustive process, but the ability to connect directly needed to be there. Ethernet was the choice, as it is a common connectivity and was easy to apply to the existing circuit board schematic, giving more time to add more features to the main device.

Further, the code lacked adequate sleep modes to ensure long battery life. By adding a sleep to the code, the system was on only for a small portion of time to check the outdoor temperatures via the sensor, then send the data to the main unit, and turn back off. No one will want to replace the batteries in their outside unit often, so energy efficiency was a must with the remote unit. A small solar-cell is included, helping relieve the strains of the battery. Several very small solar cells are available and make sense in a remote outdoor unit. Exploring the possibility of power over Ethernet was also needed.

### 1.3.2 PIC Microcontroller

Though this is the main control unit of the system, its complexity was greatly reduced. The code was trimmed, as the display will be controlled via another microcontroller (the ARM). The main PIC microcontroller needed to be able to form a connection with the secondary PIC microcontroller at timed intervals and collect the outside data. The main unit is also able to gather internal temperature and humidity readings itself, and store this information for processing by comparing with outside data, depending on user settings. The main unit also needed to be able to control the mood scents, also adjustable by user input. Vents also needed to be able to be open or closed on demand or timer via user input as well.

The user input was done through an LCD screen, so the microcontroller communicates to the LCD's controller, in our case an ARM processor. This is the most important connection for the main microcontroller.

### 1.3.3 LCD Display and ARM processor

The original HVAC unit used an enclosed LCD touch screen system with processor that just formatted images to the screen and allowed some user interaction. The system failed to implement a true user interface or have anything similar to an operating system.

A new and cheaper touch screen LCD was found that was still around 7 inches. This screen is controlled by the ARM microcontroller chosen. All user interaction is done by the touch screen. The touch screen image is controlled by the ARM microcontroller that can now use JAVA on top of an operating system to provide much more freedom of interface. The old system used C with bitmaps that severely hindered depth and added unnecessary time delays between interaction and response. On top of controlling the LCD screen, the ARM processor adds much more flexibility in connecting to the internet, either through Wi-Fi or direct connection. Through this internet connection, some form of real time display of the system over the internet is provided, through a web app. The only other processes that the ARM microcontroller must do is transmitting the user wishes to the main microcontroller and grab the data from the main controller. This is done through a direct connection between the microcontrollers using UART (RS-232) protocol.

## 1.4 Specifications

The following specifications were developed over several meetings with our sponsors as well as gaining knowledge of our device and what it can and cannot do. The unit performs all functions that the original device did. Further, the system needed to be able to accomplish functions that were installed but not implemented in the original device such as internet connectivity and mood scents. The connection to the internet provides some form of user interaction by seeing the internal temperature, humidity and CO2 data.

* 1 CO2 Sensor accurate within 100ppm
* 2 Temperature Sensors accuracy 0 to 110
* 2 Humidity Sensors accurate within 1% from 0% to 100%
* Ability to directly install a 24VAC system
* ARM LCD controller and user interface
* Secondary outside unit to measure temperature accurate as above
* Implement Operating System Timer/Scheduling for 5 Functions
* HEPA Filter and/or UV Light Control System
* Scent Disperser and Control System
* Remote Access (Http/Website, Smartphone Application, and/or Windows Application)
* Operating Temperate Range ( 0 – 100 Degrees C )
* Ventilation control
* Easily Installable and Maintainable
* Filter change notification timed
* Low Power Consumption / Energy Efficient
* Economically Advantageous

### 1.4.1 Detailed Design Features

The specifications of this project have come about through the design ideas of our sponsors. With their input, we have found ways to implement those design ideas. The unit contains all specifications of the original unit: The unit must sense temperature and humidity of the inside and outside. Then the system must make intelligent decisions on the best way to control the climate desired given the settings presented by the user.

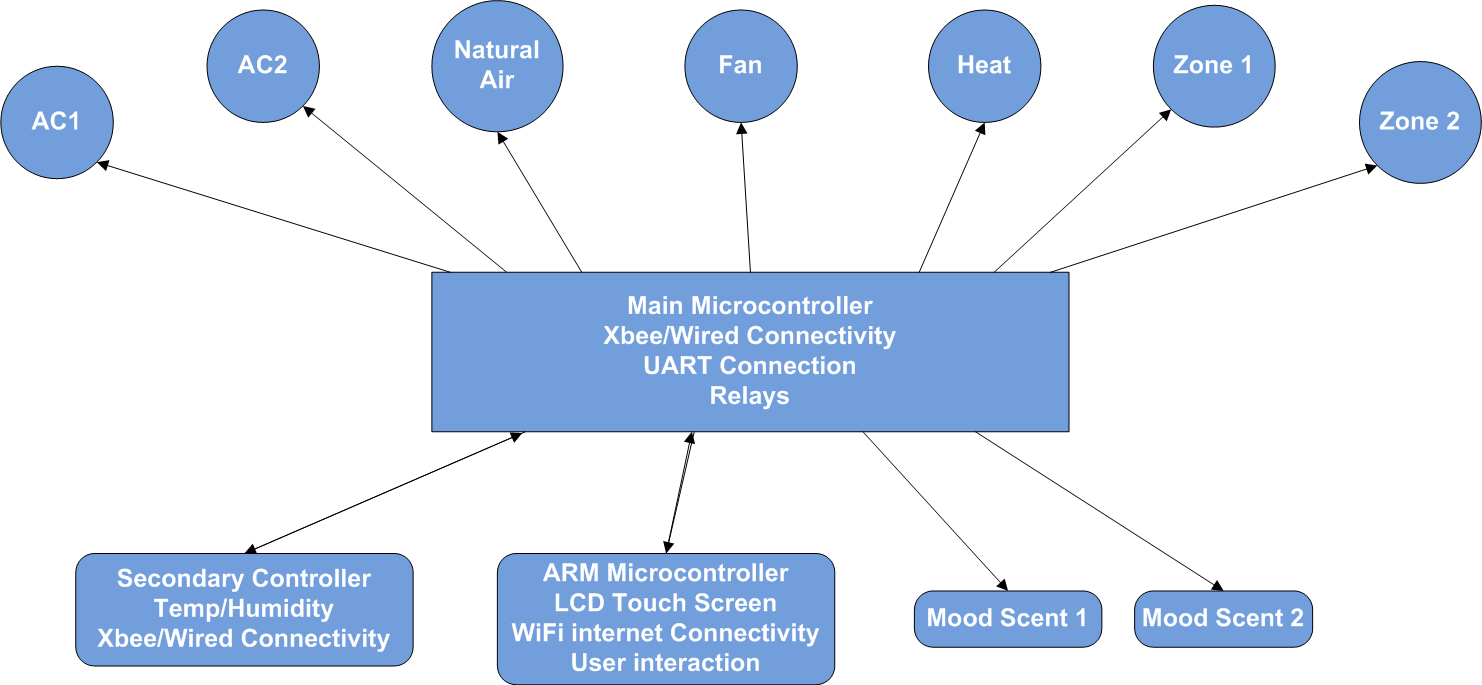
The original unit's power system was not able to link up to the houses existing 24VAC outlet as designed. The 24VAC is typical for most thermostat controllers inside of a house. A new rectifier was designed and used to power on the unit and to power the relays to turn on and off the systems air control units. Though the system air control units typically operate at 220VAC, they are controlled by the 24VAC relays in a house. Our micro-controller controls the relays.

The system needed to be able to connect to the internet. The preferred method of connection was Wi-Fi. Making a connection to a simple default setting Wi-Fi and basic remote connectivity is considered adequate at this point in the project. After making this connection, the data contained in the unit needed to be able to be presented through some web server so the user can remotely access the results thereof.

A CO2 sensor was installed on the main control unit. This is directly connected to the main microcontroller much like the temperature and humidity sensors. Carbon dioxide buildup in a house can affect the health and well-being of those inside. Typically unhealthy levels occur at 1000PPM. Future designs should consider possibly a carbon monoxide detector as Florida Statute 553.885 requires that all new buildings built after July 1, 2008, with a fossil fuel burning heater or appliance, or fireplace or attached garage, have a sensor. This is more than just a useful commodity, but it helps the safety of a household and would keep newer houses to code. As we see more and more products merge to a duality (cell phones are now cameras, GPS, music players...) the HVAC system is more than powerful enough computationally to start taking away from adding more products to the house. If dangerous pollutants are found in the house in excess, the user will be able to pre-program the unit to make and intelligent decision beyond sounding the alarm. Part of the HVAC's uniqueness is its ability to vent the outside air into the house given certain parameters. This leads to the system being intelligent enough to vent the entire house given too high pollutant level. Sleep easier knowing that the air you are breathing is being watched.

As with all newer devices, upgrades are possible. Whether it be to fix bugs, or provide more options to the user, the ability to update the system needs to be explored. Firmware upgrades can be a hectic problem, but needs to be researched. The user interfacing is done via the ARM unit, so it's possible the ability to upgrade the systems software locally could be provided. Given an internet connection, update remotely should also be at very least researched.

Installation of the mood scents was intended on the original unit, but unfortunately was never implemented. The main microcontroller needed to be able to via wired connection, control a signal to tell a mood scent device to turn on. The ARM microcontroller determines for what given time and at a given time to manipulate the mood scent device. The actual mood scent device won't be designed at this point, but a signal sent from the micro-control was needed to be sent and accurately seen working. At this point, wireless connection to a mood scent device provides remote powering problems as well as an increased cost of goods that may not be feasible in this iteration.

LCD touch interface needed to be advanced. The new unit needed to be able to have a database of times and controls of temperatures and humidity on certain days and times preset. The idea was to not require the user to have to manually adjust these settings. As the point of this system is to save costs, quantifying these results were important to the sponsors and us. The system needed to be able to quantify energy savings vs. a control and be displayable to the user to see the cost saved. A block diagram of how the system is presented below in Figure 1.

**Figure 1 Product Diagram**

## 1.5 Division of Labor

In an attempt to divide the labor intelligently and equally, we needed to assign labor with our strengths in mind. Our group consisted of one computer engineer and three electrical engineers. We focussed on separate parts geared towards our strengths in order to reach a unified goal. Dividing the labor allowed individual members to specialize and focus with expertise on their parts of the project as seen in Figure 2.

Elroy is the computer engineer in our group and was largely responsible for the programming on the ARM microcontroller. This is the device that controls the LCD touch screen. Through this touch screen is all the user interaction so it was important that Elroy’s program not only be functional, but have an interface that was easy to understand.

Elroy Ashtian

* ARM Unit programming:   
  Operating System Scheduling and Backend
* Operating System 5 Function User Interface (shared)
* Internet/Remote Access (shared)

Matt is an electrical engineer. He has networking knowledge and thus led the team in internet connectivity. He also looked into the ability to do firmware upgrades and/or software upgrades on both the main control unit and the ARM unit.

Matthew Arcuri

* Firmware and/or Software Upgrade
* General PCB design (shared)
* Internet/Remote Access (shared)

Steven is also an electrical engineer with a focus in power systems. He redesigned the power system for the whole device. Further, he had experience with Java and provided frequent help to Elroy in the user interface.

Steve Jones

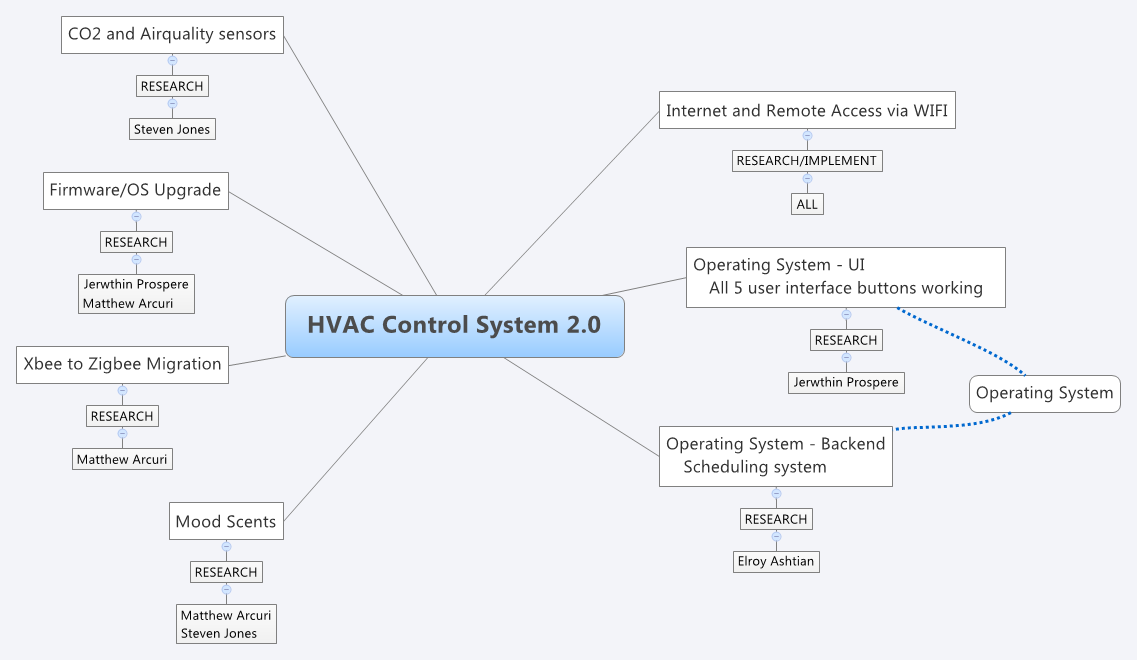
* Scent Dispenser (Displaying purposes)
* Power System (shared)
* General PCB design (shared)

Jerthwin is also another electrical engineer. He had a very strong background in microcontrollers and led the team on programming of the PIC microcontrollers and as well in dealing with the connectivity between these devices.

Jerthwin Prospere

* PIC Microcontroller Programming
* General PCB design(shared)
* Power System (shared)

As seen in Figure 2, we will all tackled the internet protocol and will all helped build every device as time is permitted. Jerthwin and Elroy did the majority of the coding while Matthew and Steven supplied the power and provided connectivity to all the sensors going through the system.

****

**Figure 2 Division of Labor**

### 1.5.1 Coding

Although the majority of the products designed were implemented in the second stage of the project, it was important to divide this up evenly. Elroy was the computer engineer in our group, but we couldn’t solely rely on him to do all the coding in this project. Elroy was in charge of programming the main operating system on the ARM microcontroller. This was done in JAVA which Steven has some experience in and lent a helping hand to Elroy. The main microcontroller implemented code similar to the first device, but was plagued with errors and commented out code. The code needed to be refined and better implemented. Further, the microcontroller needed to be programmed to talk to the new CO2 sensor as well as control the mood scents and vents. The majority of this programming was done by Jerthwin. How the two processors talk to each other was a joint effort, tackled by all group members. Mathew has insight in transmitting the data over the internet and was leaned on heavily for the coding of the internet as well as figuring out the best way to implement future upgrades to the system. As we attempted in the division of labor, division of coding responsibilities were partitioned out to our own individual strengths. Table 1 illustrates the specifics of where each individual coding responsibility was.

|  |  |
| --- | --- |
| **Group member** | **Responsibility** |
| Elroy Ashtian | ARM Microcontroller and main operating system. |
| Jerthwin Prospere | PIC microcontroller. PIC microcontroller interconnection. |
| Steven Jones | PIC microcontroller connection. |
| Mathew Arcuri | ARM Internet connectivity and remote access. Firmware and/or software upgrades. |

**Table 1 Code Responsibility**

### 1.5.2 Milestones

­­­We decided that it was best to meet frequently in the senior design lab to work on our research and design. We wanted to get together as often as possible to get used to working together and provide a sense of camaraderie. Further, meeting in the senior design lab helped get us familiar with the location where the actual system will get put together and tested. Our product was compiled and put together using windows live. We started our project with a table of contents and portioned out our labor evenly according to it. As sections of the table of contents got complete, we compiled the paper together and uniformly. Each section was edited by another member and added to, in order to provide a continuous flow of thought.

#### 1.5.2.1 Senior Design I

We were the second group to choose projects and met three times the week before to discuss sponsored and non-sponsored projects. This project was at the top of our list and we were fortunate to get it. After meeting with Dr. Richie and getting approval for doing this project, we set up a list of milestones for us to reach. These helped us reach the goals necessary to finish this project. We looked at layouts of multiple groups before to help us understand proper formatting. We wanted to not have to rush the project at the end but do research and design as we went and write accurate and good papers on such. Figure 3 is a timeline for our senior design 1, in which entered and complete the research and design portion of our project.



**Figure 3 Milestone Sr. Des. 1**

* Project Assignment was the last week of January when we were assigned to find a project. We wanted to make a decision immediately. We didn’t want to go back and forth debating on which project to do.
* Organization was the week immediately of and following our project assignment at the end of January. The purpose of the organization was to develop milestones and divide up our labor. We wrote a simple table of contents for our project containing all the parts we felt at the time were necessary to complete it and portioned off section of the table of contents for individuals to write. The act of writing became our research.
* Research started immediately after organization and provided the largest portion of our semester’s milestone in the end. While we had opportunity to experiment with the original unit, we had to figure out how it worked and failed. From here we worked on fixing the original prototype to better understand the project. At this point a complete break off was made from the first unit to make our own original device learning from the mistakes of our peers.
* Parts research and acquisition went beyond the broad idea of the system and focused on the individual areas and how to implement them with real life parts.
* Design was done as soon as part research was completed. This wasn’t just where we learned how to make the individual parts of the project, but learned how to interface them together in a complete system that was workable. Making the individual parts was worthless without seamless integration into a complete working system.

#### 1.5.2.2 Senior Design II

Senior design II was where we implemented our design into a working device. All of our research was worthless if we couldn’t put all of the pieces together into a working prototype that met the criteria set forth in our design specs. This portion was during the Summer semester, giving us a few less weeks than usual, so we needed to have our entire design finished before the start. We met as often as possible together in the Senior Design labs, as we’ve all already garnered access keys. The plan was to stay focused and have parts ordered well in advance and the device tested with enough time to make significant if necessary modifications. The milestone for senior design II is shown in Figure 4.



**Figure 4 Milestone Sr. Des 2**

* Develop a working prototype with all the given specs from above that was researched in senior design one and found to be feasible.
* Testing and debugging of system in order to feel confident in its presentation and ability to replicate.
* Presentation of system

## 1.6 Financing

Our sponsor of the HVAC system agreed to fund the system up to $1500. If parts exceed this price but are necessary the sponsor provided encouraging words of flexibility on this number. While the majority of this project was built using parts of insignificant cost, a select few of the devices provided the majority of the cost share. Implicitly implied here are the LCD touch screen and its microcontroller.

Some of our parts were ordered by our sponsor and shipped to them. Once we realized that it would be more convenient to order the parts ourselves, we settled with the sponsors to have them refund us. We did feel at the beginning that the project would go over budget, but because of the additional parts and quick shipping, we went over it. Having provided us with a development board for the microcontroller, it helped us recede on our budget.

# 2 Research

The purpose of research is to investigate all things in our project to be implemented or desired. Additional items will be added in research that may never make it to design as feasibility and keeping the project within scope of time to finish as well as under budget will become factors when deciding which items need to go to design. When it is unknown which part or feasibility, the section will be split amongst two or more members and it will be equally researched at different angles and upon conclusion at a meeting amongst members decisions will be made on the product at hand.

## 2.1 Research Methods

Our group met at least once a week in person to discuss where and assign different portions of research. We tried to split up each area to our specified areas of expertise. In areas where decisions had to be made on parts, we divided the research into two or more parts and let the research lead us to the correct parts.

In order to fully understand the system, the research started with the old unit to investigate its successes and failures. We spent several weeks looking at the physical device and its software. After we fully understood how the original device functioned we broke off in the areas that were either new to the system, or fixes to the system to decide on parts that we would keep in the device or replace.

Several meetings were set up with our sponsor to plan and give us feedback as into the importance of our research. It was important for us to add several key features but to not implement too many new ideas that we would not be able to accomplish the important ones as effectively and usable as possible. It was determined by our sponsors that the user interface and actual functionality of the unit to implement correct and intelligent decisions on dealing with the HVAC system of the house was by far the most important features. The added details of mood scents, Wi-Fi connectivity, zones, and a CO2 sensor are important and need to be added, but are irrelevant if the user interface fails to be implemented in a proper fashion. From these meetings we have set a priority of our research and research to lead to development on the programming of LCD screen software.

## 2.2 High Level Control Unit

The most user interactive component in our system is the high level controller. The high level controller will be the one of the most complex parts of the system on both the software and hardware side. The high level controller will be located in the user interface control unit and is where the data received from the primary slave controller will be processed for distribution to the user. It is also where user interaction meets the data presented from the components in the entire Efficient HVAC Control and Feedback System. The high level controller is required communicate with a IEEE 802.11b/g Wi-Fi Module, a LCD touch screen, and the primary slave controller.

Internally, the controller must be programmed to utilize the data shared by these components to provide accurate readings and commands through the system. The programming for this component will be adaptable and very expressive to provide substantial data to the users and interconnected subsystems. Also, the high level controller must be flexible enough to handle program changes throughout its lifecycle that may be added in the future to incorporate additional features that may be added to the Efficient HVAC Control and Feedback System.

Because the high level controller is required to interface with other components, it must have a sufficient amount of memory, processor speed and designated ports such as I2C, SPI, and UART to be able to accommodate the communication requirements of the other components. Before a high level controller could be chosen, the requirements and specifications of our project had to be considered, and a choice had to be made that provided a controller that had the necessary components to complete the project.

Since the high level controller is the most user interactive component of the entire system, it must be responsive and display useful data whether in a perfect working condition or even in a condition of where partial or all components fail. The development of the high level controller will be one of the most important aspects of the overall project build and everyone in the group will be involved. We will be in close communication with the sponsors prior to beginning the development of the high level controller to ensure we develop the controller to operate to their exact specifications and goals. The performance of the high level controller is truly a critical aspect of the Efficient HVAC Control and Feedback System that we are developing. The section below describes the research conducted to find the most appropriate controller for the given set of requirements. The following are some topics that need to be addressed:

The high level controller interfaces with a LCD touch screen, and a IEEE 802.11b/g compliant Wi-Fi Module. The high level controller chosen to work with was the Beagle Board. Since the high level controller is developed by Texas Instruments, Inc and sold by DigiKey, we decided to start our search for the necessary components there. All of our components will be placed on the chosen board to keep our design as space efficient as possible. The high level controller will be small and only account for a small percentage of the overall design of the project.

When exploring controller options, we have found that most controllers have at least one internal analog to digital converter. A converter may be needed when communicating with the wireless router that will connect the system to the internet. Therefore, at this point we are only considering controllers that have at least one analog to digital converter on the board.

The development board we purchase should be the least expensive model that meets the necessary requirements of our project. Most boards within the class that we are going to implement should be $400.00 or less and so we would like to stay within the $0 - $400 range. Being between $0 and $400 dollars, the high level controller is only an essential percentage of the total cost of the system although it is the one of the most important components.

### 2.2.1 ARM Microprocessor Based Master Main Controller

The main component of our product will be the ARM processor that is located on the Main Control Unit. This processor will be used to communicate to all the devices such as LCD screen, relays, CO2 sensor and the slave PIC processors. The ARM will be the most complex part of the system both hardware and software perspective. There will be a lot of effort to create the software to effectively do all the tasks that is required from communication to User interface control. In the hardware perspective, we need to have a robust design that will enable us to provide this processor with its necessary voltage and current. Traditionally, ARM processors are low energy efficient which puts us in the right position in terms of energy efficiency. Due to this characteristic, you will most likely find this type of processor in mobile devices, digital media and music players, hand-held game consoles, calculators and computer peripherals such as hard drives and routers.

The ARM processor is a 32 bit RISC developed by ARM Holdings. Based on the insight, RISC can provide higher performance since there is a much faster execution of each instruction. This is particularly attractive for us, since we are looking to have a system that is user friendly and attractive. We find in the previous model many bugs which causes the system to run very slow. That could be really a pain to the user having to press a command multiple times due to the lag that is experienced. We would like to be able to give the user the comfort of immediate response from the system. It is imperative that we achieve this for this will enable the marketability of the product. Once again, the ARM processor having RISC characteristics is very attractive to us as engineers working on this product. Also, having 32 bits instruction width will ease decoding and pipelining at the cost of decreased code density. Hence, we see that the time is optimized by this machine. Truly, achieving our goal of fast response to the customer is reachable based on the attributes of this processor.

The method of programming of this processor type is also maximized to provide complete speed efficiency which is one of the main upgrades we are looking to perform from the first prototype. The ARM architecture includes features to improve code density and performance through: instructions that combine a shift with an arithmetic or logical operation, auto-increment and auto-decrementing addressing modes to optimize program loops, load and store multiple instructions to maximize data throughput and conditional execution of almost all instructions to maximize execution throughput. Having these features enable the ARM processor to attain a great balance of little code size, energy efficiency and most of all an augmentation of performance as opposed to version 1. The ARM coding language is that of C which is ideal for all of us, since we are all familiar with the C language. C will be the language used to do all the hardware objectives, whereas, JAVA will be used to implement the user interface. One of the great aspects of the ARM processor is that we will be able to import Linux onto the processor which will enable us to compile and use JAVA to do all the user interface objectives done by our Computer Engineer. We are aiming to have a processor that will enable us to have a sophisticated user interface and a robust hardware design to facilitate an unequivocal goal of high performance.

Since we already have the first version of the system, we decided to opt into having the ARM processor as a master to the PIC processors that are already there functioning. Instead of reinventing the wheel, we have decided to keep the previous design of the PICs and add the ARM as the master simply because the ARM will enable us to have great flexibility in creating a sophisticated user interface, increase system performance and give us exposure to the current industry standards. Using the ARM as master would leave the PIC to just receive data from the temperature and humidity sensors and report it to the ARM when it has to. The ARM will then do the task of switching the 24V control relays to open or close, control the user interface, attain wireless connectivity via 802.11b transceiver and communicate with the PIC processor.

The ARM internally must be programmed to use the input readings from the PIC processor and sensors to make the correct decision about how it will choose to cool, heat or ventilate the building. The programming needed to implement this task will require much memory which is why it is important that the appropriate level of memory is picked to facilitate this. There must also be enough flexible such that code can be updated in the future to incorporate new features to the Efficient HVAC Control and Feedback System.

Due to the fact that we have to interface the ARM processor with so many other components, it must have sufficient amount of output and input pins and designated ports such as I2C, SPI and UART to be able to meet communication requirements. Before we make any decision on a microcontroller, we have to take into account the specs and requirements for our project effectively. The fact that this is the brain of our entire system, it must function perfectly for the system to work. It will be the responsibility of the entire group to ensure that this part of the project is functioning according to the desire but Jerthwin will have the main task of putting it all together. We will be constantly communicating with the sponsor to ensure that we are as close to their goal as possible. The performance of the ARM is definitely the defining point of the Efficient HVAC Control and Feedback System as it will determine the performance of the entire system. There are many complexities in choosing the processor which will be discussed in the next few sections.

#### 2.2.1.1 Processor Compatibility with Slave and other components

The ARM must interface with a PIC processor, LCD screen, an 802.11 wireless Wi-Fi chip and CO2 sensor. We have a little concern with the ARM and PIC being compatible which led us to multiple websites and conversations with advisors on the internet. We came to conclusion that SPI is a communication protocol that is a basic standard. We could use this protocol to communicate between the ARM and PIC. The ARM processor will have a much higher clock frequency compared to that of the PIC’s will be slower than the ARM, therefore the master will first configure the clock using a frequency less than or equal to the maximum frequency the slave supports. Based on our research, we could easily interface the existing LCD screen to the ARM via RS232 or possibly UART. With a new LCD screen we could use the more standard DVI/HDMI ports. There will not be any problem interfacing the CO2 sensor to the ARM.

#### 2.2.1.2 Constraints on size of Microcontroller

All the components of our system will be place on printed circuit boards designed by ourselves except, at least for this version of the prototype, for the ARM microcontroller. We have to be wise in the selection of our parts as size, cost and power efficiency will be critical to our design. In order to keep our PCB relatively small, we will choose the smallest parts available which are usually surface mount components. The PIC processor is relatively small so we shouldn’t have an issue in terms of size when it comes to our processors.

#### 2.2.1.3 Analog to Digital Conversion and Input/Output ports

There are many options for the ARM processor to have the ability to convert analog signals to digital data. This will be the key for us because we are going to be interfacing a CO2 sensor to the ARM. In terms of the other readings, the PIC(slave) will be able to do the conversion and would therefore just send the ARM the digitalized data. This could also be of importance in the aspect of wireless connectivity. We are left to no choice but to ensure that we have chosen an ARM processor that has at least one analog to digital converter. The Input / Output ports are how the processor interfaces to the other components of the system. For this version of the project, the ARM will have to connect to the PIC which interfaces with the 24 V control relays, 802.11 (Wi-Fi) transceiver and the CO2 sensor. The need for I/O pins is heavily demanded by the relays which requires quite a bit since they operate the other components of the HVAC system such as AC1, AC2, Dehumidifier, Mood Scents, compressors etc. The sponsor had stated to the previous team that they would like 20 I/O ports available specifically for control relays associated with the HVAC system which we plan to also meet. Having this stipulation definitely means that we have to choose a slave PIC processor that has more than 30 I/O ports. According to the research done so far, a lot of the PIC processors have an I/O voltage supply of 3.3 V which is what the current product gets out of the PIC processor used by the previous team. If we decide to use the same relays that they used, then we would not have a problem with output voltage. We are definitely allowing some flexibility to exist with the number of I/O ports that are available to use as we may have the opportunity to add additional features to the system.

#### 2.2.1.4 Communication Protocols / Integrated Peripherals

Since there is a lot of communication that will be done in the system, we need to ensure that we have enough channels to support I2C, SPI and UART. We know for sure that we are going to communicate to our LCD via the UART. The SPI will be used to communicate to the PIC(slave) processor. We need to determine the amount of I2C channels we need based on the number of additional sensors we add to the system. For now, we will definitely need it to communicate to the CO2 sensor and the 802.11b wireless chip.

#### 2.2.1.5 Memory Type

We are going to consider processors with Flash memory for our ARM. Flash memory is a type of EEPROM (electrically-erasable programmable read-only memory) that is non-volatile storage technology that can easily be electronically erased and reprogrammed. Using flash memory in the processor will give us the flexibility for changes made to the system by the sponsors after the prototyping phase. Based on the extensive research done, the majority of the ARMs come with flash memory which is also much more than the amount version 1 has. Another issue relating to memory is the amount of RAM that we will be able to utilize. Based on the PIC processor used by the previous team, they had 30 KB of RAM which could be a reason why the system has such an annoying lag with input on the touch screen. Furthermore programming to include a good user interface and functional Wi-Fi would’ve had to been done with this limited amount of memory. In our position, having the ARM driving a real modern operating system can give us the luxury of having at least 256 megabytes (MB) of RAM which would make our system run a lot faster especially considering that we will be running Linux to support of user interface which is a light weight operating system. The sponsors have made it apparent that this product will endure be evolving as more iterations of prototypes and production versions are done which gives us ever more the reason to choose ARM which is powerful embedded platform with a rapidly decreasing cost. The ARM boards we are considering also have the flexibility of changing the ROM being removable flash which makes rapid prototyping and upgrades easier.

#### 2.2.1.6 Cost

The developmental board with the ARM processor we purchase should be the least expensive model that meets the necessary requirements of our project. This still however will be a small percentage of our total budget. The difference between version 1 and version 2 of course is the ARM processor and the main PIC controller we can substitute it with a PIC similar to the one on the Secondary Unit. It is to our benefit that we already have in our possession the development board for the PIC processors from the previous group. We decided to choose the BeagleBoard from TI as the vendor for our ARM processor and developmental board. We will discuss this decision at length later.

#### 2.2.1.7 Voltage and current requirements

The source voltage that will be supplying our main control unit is assumed to be 24 V AC. This assumption has been made based on the fact that the main control unit will be installed where there was previously a thermostat mounted or installed in a new HVAC system as the main thermostat. Most thermostats in HVAC systems are powered by a 24 V AC wire that is installed when the building is built. The processors that we have been looking at typically require an operating voltage between 1.5 V and 3.5 V. This voltage will have to be generated on the printed circuit board which will be discussed in the PCB Design section of the paper.

The processor will be the work horse of the system and will need to be able to do the majority of the processing, making crucial decisions and sending and receiving all information. This is the brain of the system as it was stated earlier with help from the slave PICs. It is imperative that we keep the voltage and current spec as this will determine the performance of our entire system. Much detail will be placed on power supply design as this was one of the downfalls of the previous team.

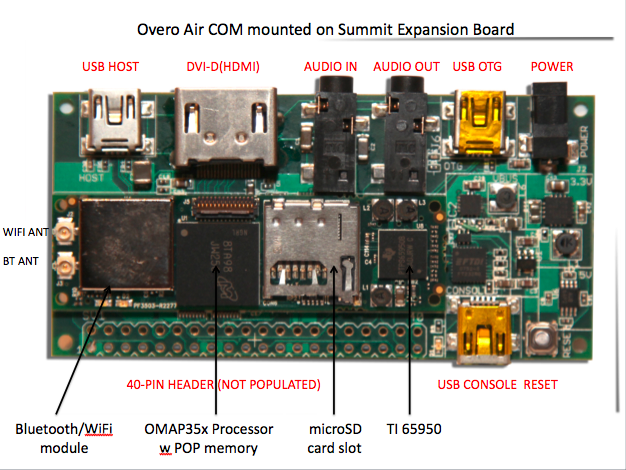
It is also important that we have the correct I/O voltage supply for the components that we are interfacing to such as the sensors. Generally, sensors don’t have high operating voltage demand which gives us the comfort in selecting the ARM processor as our master processor. As previously stated, most of the ARMs have an I/O voltage supply of 3.3 V which should be sufficient to drive the sensors and chips that we will interface to the ARM.

### 2.2.2 High Level Control Unit

In order to provide a more interactive and friendly user experience, we programmed the user interface using JAVA. JAVA is a high level language, requiring an operating system. The following units will serve as the base for minicomputer base to run the operating system and program.

#### 2.2.2.1 Gumstix Overo

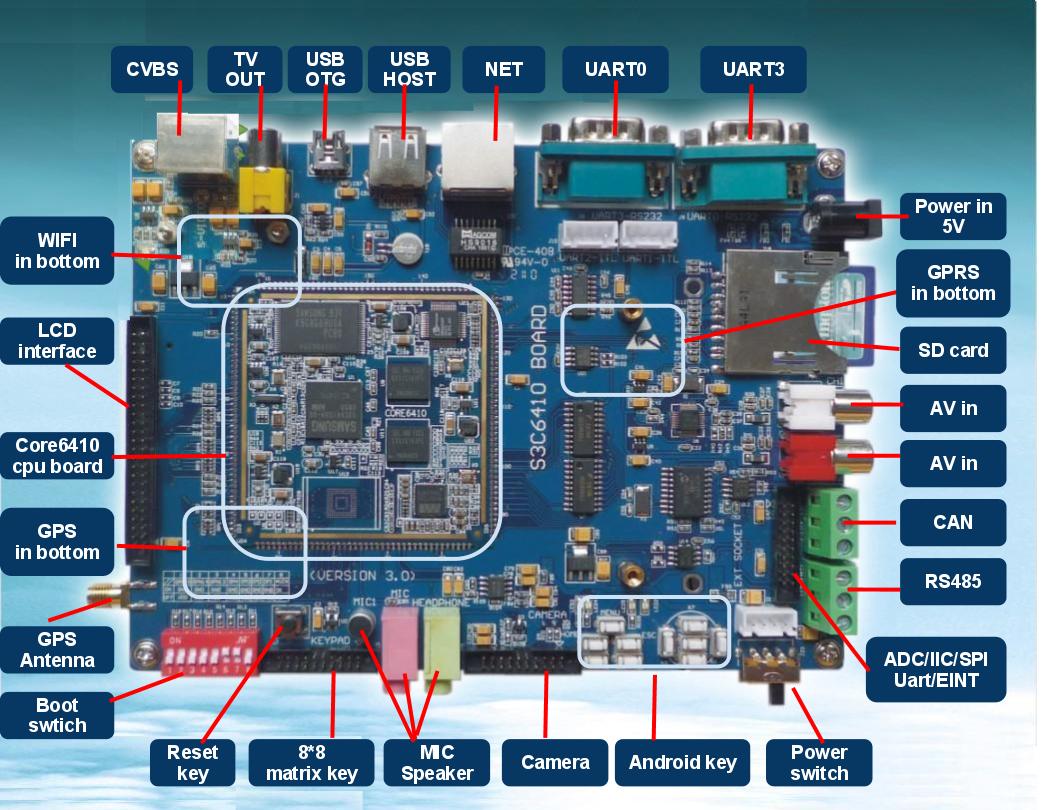
The first proposed development kit was the Overo Air EVM pack which includes the Overo Air Com, as see in figure 5 below. It is produced by Gumstix, Inc. The pack supplies an Open Multimedia Application Platform (OMAP) 3530 applications processor, Bluetooth and 802.11(b/g) wireless communications, high speed USB Host & On-the-Go(OTG), 8GB SD storage and a LG 3.5” touch screen LCD display. The OMAP processor included is a customization of the ARM Cortex-A8 Central Processing Unit (CPU) architecture which has a frequency of 720MHz which can handle up to 1200 Dhrystone million instructions per second (MIPS) , a C64x+ digital signal processor (DSP) core and also contains a POWERVR SGX Graphics Processing Unit (GPU) that allows for 2D and 3D acceleration. The memory capacity is 256MB of Random Access Memory (RAM) and 256MB of Flash memory. The kit also has a I2C port, 6 Pulse-Width-Modulation (PWM) lines, 6 Analog/Digital (A/D) input lines, 1-wire, UART, SPI, Extra MMC lines. This kit fulfills most of the requirements for a high level controller but the two major shortcomings are with the board’s inability to provide a Serial port that would be used to communicate with the main controller and the size of the screen which is under the desired size of 7”. The complete cost of the kit is $442.00 which is over our expected expenditure for the board. Below in Figure 5 is the useful ports and overhead diagram of this unit.



**Figure 5 Overo Air Com  
(Reprinting Permission requested from Gumstix)**

#### 2.2.2.2 Linksprite

The development tool that we selected is the LS6410 S3C6410 ARM11 Android Development Kit. It is developed by Linksprite Technologies, Inc. The Development Kit includes a 7" Thin Film Transistor (TFT) LCD Screen, a S3C6410 ARM11 Development Board, a 12V 2A power supply, a serial cable, USB cable, Ethernet cable and a DVD-ROM with product reference. The development board contains a S3C6410 ARM11 Core Board that integrates key components such as a Samsung S3C6410 mobile processor, two 16 bit 128MB mobile Double Data Rate (DDR) RAM, a 1 GB Multi-level Cell (MLC) NAND Flash, power management circuit, and Ethernet chip. The Development Board is designed to easily to communicate with a computer via USB or RS232 interface. This board is ideal for our project because it can be loaded with Linux which would allow the ability to utilize an operating system for the various features of our software application which will be written in the Java programming language. The Java programming language is a high level language which makes developing a User Interface for the users to interact with the system easier. Java is a language that we all have experience with as a group, and therefore this is makes our programming needs obtainable. The price of this development is $279.00 and requires an additional $45.00 for the IEEE 802.11 b/g Wireless Local Area Network (LAN) module. The overall cost for this board is within our allocated expected range of spending. The only issue with this board is the unknown. All of the other boards reviewed are commonly used and have active communities on the internets. Customer support and lack of supporting data will weigh heavily in the decision of the board in the end. The company selling the board was contacted on this matter and we were reassured that although the forum community is not the most active for this board, they have a very good customer support staff that is more than willing to help us obtain the goals we need. The board and its ports can be seen below in Figure 6.

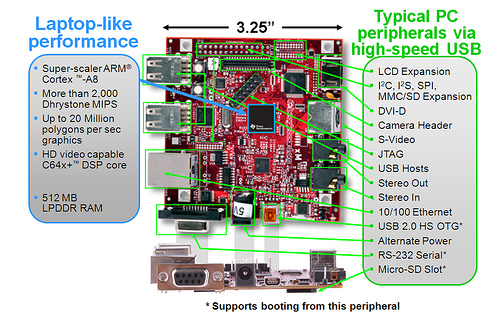
****

**Figure 6 LS6410 ARM11 Android Dev Kit**

**(Permission Requested from LinkSprite)**

2.2.2.3 Beagleboard

The third proposed board(s) are the Beagleboard-xM and the Beagleboard revision C4 produced by Texas Instrument. The board’s key components provided are a TI ARM Cortex A8 processor, high speed USB OTG and SD Card Slot. The OMAP processor on this board has a clock frequency of 1GHZ MHz and has similar features as the Gumstix kit. The memory on the board is 512MB Low Power Double Data Rate (LPDDR) RAM however comes with 0MB NAND flash memory. It also comes with a 4GB Mirco-SD flash memory card and is capable of High capacity SD flash cards 32GB or higher. The board is also capable of supporting the following communication interfaces 10/100 Ethernet, USB, RS-232 Serial, I2C, UART, and SPI. Both the BeagleBoard C4 and the newer BeagleBoard-xM model are both in production. The BeagleBoard-xM generally has better specifications and more features however the biggest drawback is the lack of an NAND flash onboard. However is compensated by the flash card size being 4GB. A comparison of the BeagleBoard Revision C4 and BeagleBoard-xM follows in figure 5. Row regions highlighted yellow show where one revision has an advantage over the other revision. Rows regions with no highlighting are features that are not important for our design or where neither of the boards have a significant advantage. Figure 7 below is the top view with ports of the BeagleBoard-xM



**Figure 7 Beagleboad**

(reprinted with permission under the Creative Commons License)

Both boards have the performance to accomplish the tasks deemed for the specifications needed of our high level controller such as I2C, UART, RS-232, USB, and a DVI/HDMI port. Both these units have processor speeds near or at 1GHz and memory 256MB or higher will mean that the responsiveness of the unit to the user will be more than adequate on a Linux based operating system with the processor cycle workload our tasks will use, even if we were to implement our software in the resource intensive JAVA programming language. The one lacking feature of both of these united, compared to prospective boards 1 and 2 is built in Wi-Fi. This required specification can easily be filled via a USB attached Wi-Fi module. While we are somewhat concerned about possible driver difficulties caused by a USB Wi-Fi module to meet this requirement based on our experience with drivers on Linux in the past and the notoriously poor quality of drivers for USB Wi-Fi devices, we feel given proper research of a module we will be able to avoid such incompatibilities. Both board significant specs are in Table 2 below.

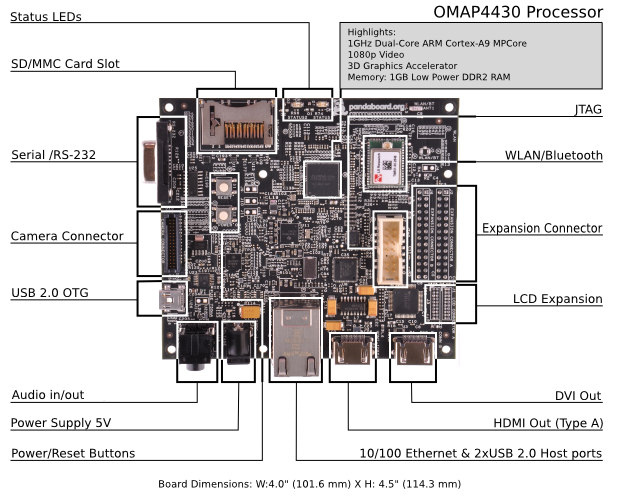
|  |  |  |  |
| --- | --- | --- | --- |
| **AREA** | **-xM** | **C4** | **Note** |
| Processor | DM3730 | OMAP3530 |  |
| ARM Frequency | 1GHz | 720MHz |  |
| DSP Frequency | 800Mhz | 520MHz |  |
| DDR | 512MB | 512MB |  |
| DDR Speed | 166MHz | 166MHz |  |
| NAND | 0 | 256MB |  |
| SD Connector | uSD | MMC/SD |  |
| USB Host Ports | 4 | 1 |  |
| Host Port Speed | FS/LS/HS | HS |  |
| Serial Connector | DB9 | Header | Direct connect to USB to Serial Cable |
| Camera Header | Yes | No |  |
| Included 4G SD | Yes | No |  |
| Overvoltage Protection | Yes | No |  |
| Power LED turnoff | Yes | No |  |
| Serial Port Power Turnoff | Yes | No |  |
| MMC3 Expansion Header | Yes | No |  |
| McBSP2 Expansion  Header | Yes | No |  |

**Table 2 Beagleboard Revisions**

2.2.2.4 Pandaboard

The last board we extensively researched was the Pandaboard. The PandaBoard seems to be the successor to the BeagleBoard and in many ways it is superior. Priced at just $25 higher than the Beagleboard-xM, the PandaBoard features the best of performance and features per dollar spent, in other words the best cost to performance. The PandaBoard features an OMAP4430 Processor which is a 1GHz Dual-Core ARM processor. It also features 1GB of DDR2 RAM, accelerated graphics capable of driving 2 HDMI displays and can output 1080p – or 1920x1080 resolution – video. These are all features that are improvements over the BeagleBoard-xM specifications. It also features built in Bluetooth and Wi-Fi internet connectivity. As seen below in Figure 8, the PandaBoard provides a more powerful board with more connectivity options. With the additional cost of a Wi-Fi dongle the PandaBoard actually come out ahead in terms of cost compared to the BeagleBoard-xM and offers a better development environment as the additional processor specifications all for faster rapid prototyping on both software and firmware.

While the PandaBoard does not include a 4GB flash card that the BeagleBoard-xM does include, the size of the flash drive is too small for our needs anyway and is cheap flash card with slow read/write times compared to flash cards available needed for digital video and picture recording in camcorders and camera. This translates to a slow system response for the user and for rapid prototyping, should we need to rebuild the operating system to fix incompatible drivers the time saved is enormous. This will likely be an issue as the PandaBoard at the time of this writing does not have drivers for the WiFi or graphics modules included in their development ports of the basic Linux, Android, and Ubuntu distributions that provide. We anticipate saving a lot of time building and rebuilding our Linux environment with a faster flash drive. The price for a class 10 – which is the fastest speed rating of a flash drive available – is not significantly more than $30, and possibly even less. So with the clear winner in terms of price, performance (for fast prototyping of software and OS builds), and features is the PandaBoard. Unfortunately at the time of this writing it is not available for immediate shipping. The main supplier, Digikey, has a backorder of 2500 boards. We placed an order for a unit anyway, in case it comes in time for us to be able to use for our project. With that in mind we will pick a second best from the other boards that we can get immediately.



**Figure 8 Pandaboard  
(permission granted under the Creative Commons License)**

Upon comparison, despite the BeagleBoard Revision C4 retailing at $25 cheaper than the BeagleBoard-xM and the BeagleBoard-xM performance more than exceeding what we need, we feel the BeagleBoard-xM – with the higher clock specifications and larger RAM – will offer a much better user experience for the extra $25 cost of the part over the older generation BeagleBoard Revision C4. The BeagleBoard-xM offers better features than that of the previous mentioned prospective boards in that it is an all in one design with large community support. The Gumstix – Prospective Board #1 – has various modules that need to be purchased for certain features and isn’t as powerful of a device compared to also doesn’t allow us the possibility of using an RS-232 to fall back on for communication with the PIC microcontroller. The Linksprite produced board – Prospective Board #2 – also didn’t have as good community support as we would like and we would’ve had to compile Linux from scratch if we didn’t use their in house compiled Linux distribution, which seemed to be lacking a lot based on their user manual. Upon reviewing the installation procedure for the Linksprite board, it was clear that the process of having a distribution for their board had been poorly outsourced and wouldn’t have the capabilities of Ubuntu and other full featured Linux distributions that both the BeagleBoard and Gumstix.

## 2.3 Microcontroller as Slave

Our main control unit will gather the code from the ARM controller and send these instructions to the actual units via relays. The microcontroller on this main control unit will be the slave of the ARM, administering the commands of the ARM unit’s logic. Further, it will gather the technical data necessary for the display unit to make the necessary logical decisions.

### 2.3.1 PIC

The microchip is going to remain as the computational device for all of our measurements. In our project, we are stripping the role of any display for the PIC and assigning it to the ARM processor. The PIC will therefore act as a slave to the ARM where it will be providing information to the ARM when it is needed to. Based on our deliberation, the PIC will be connected to the sensors, WIFI modules, Relays and the secondary unit. We have decided to keep the dsPIC33FJ256GP710A as the processor to supply the information to the ARM. According to the previous teams’ and our research, this processor allows us the flexibility for upgrades which is of importance to the sponsor. At a hardware perspective, there isn’t any major configuration to do to the ARM with any upgrades. The main upgrading will be done in establishing a solid embedded system to efficiently give the additional information to the ARM. We have also decided on using UART as our official communication method to implement the slave and master configuration. Using this communication style will ensure that we don’t have much lag and give us reliability in the receipt of information. This method was chosen because we would be able to use a cable to establish a secure connection. If we were to use any other style, we would have to solder wires to the ports, which would make our hardware a bit of a mess.

This PIC will be the key to having an accurate system and therefore we are putting an emphasis in ensuring that it is performing to par. The PIC processor will be once again the hardware responsible for all of our readings which include temperature, humidity and carbon dioxide. It is required to accept input from a Zigbee wireless transceiver, a temperature and relative humidity sensor (also located in the main control unit). Compared to version 1, this PIC will not be connecting to the 802.11b chip nor the LCD touch screen. Both of these tasks are now going to be a part of the ARM processor. We made the decision to so because we believe that it will be easier to establish an internet connectivity with the ARM due to the fact that we can run an actual operating system (Linux). In the previous version, this processor did the entire decision making in terms of how to cool, heat or ventilate the building, but we have chosen to do that with the ARM processor. The fact that we have done this means that we don’t have the need for ample memory.

Due to the fact that the PIC is required to interface to so many components, it must have sufficient amount of input and output pins and designated ports such as I2C, SPI and UART to be able to cater to the requirements of other components. Before settling with this workhorse, we had to look at the specs of the project and the upgrades that were desired by the sponsors. The choice was made that we go with this one because it would take us out of the situation of reinventing the wheel. So, rather than spending the time to start the project from scratch, we can now build upon what the previous group did.

Our main concern is that the user will be getting real-time data without much delay which would require us to synchronize the PIC(slave) to the ARM(master) is such a way that the data being display to the screen is accurate. The timing between these two will be very important as this is what is going to be used to make decisions. We will continually be in contact with the sponsor to ensure that we are giving them the features that they are investing in. The interaction between the PIC and the ARM is definitely a make or break aspect of the Efficient HVAC Control and Feedback System that we are developing. The research done will be explained in the following sections that enable to make the decision as to stay with the PIC microcontroller as our computation device.

#### 2.3.1.1 Microcontroller Compatibility with other Components

The PIC must be interface with a XBee wireless chip and a Sensirion SHT21 temperature and humidity sensor. Since version 1 had no problems using these parts, we decided that it would be appropriate to use them likewise. The previous group started off trying to use the Zigbee wireless, which is made by Microchip, but because of the difficulty that they experienced, we have decided to divert from using it. Because of the fact that we are doing our wireless connectivity with the ARM, there is no need to worry about the compatibility with the 802.11b chip.

#### 2.3.1.2 Analog to digital and digital to analog converters I/O

At this point in technology, most microcontrollers have the ability to convert between analog and digital. We will definitely need this capability as we will be communicating wireless between the PIC processors. Of course, we won’t be able to communicate wirelessly using the analog signal, therefore, we need to have the ability to massage digital data.

The input and output ports will allow us to communicate to the outside world with our PIC microcontroller. For our version of the project, the PIC must communicate with a temperature sensor and humidity sensor (2 I/O ports), a XBee wireless chip (4 I/O ports) and to the ARM processor (3 I/O ports). This comes to a total of 9 I/O ports for the PIC to communicate with the components on the main board. In addition to these components, the PIC is going to have to connect to relays that control the other components of the HVAC system such as compressor, air handlers, and a dehumidifier. The sponsors have specified that they would like to have 20 I/O ports available specifically for controlling relays associated with components of the HVAC system. That now brings our total I/O ports to 29.

The previous group was able to do the necessary research to bring them to the decision of the Microchip dsPIC33FJ256GP710A which we were in complete accordance with. After attempting some development, we switched to the PIC24FJ128GA010 for the main board due to some silicon errata and to provide continuity from the secondary and main control unit. This microcontroller meets the ports requirements and contains even more for upgrades. The possible add-ons to the system are air purification and scent emitting devices. The upgrades that are taking place will definitely require more software design and implementation.

#### 2.3.1.3 Memory Type

Since this microcontroller is just taking in information and then sending it to the ARM, we are not in important need for lots of memory. Due to the fact that we would still need memory to implement our design, we will go with flash memory. Flash memory is a type of EEPROM (electrically-erasable programmable read-only memory) that is a non-volatile storage technology that can be easily electronically erased and reprogrammed. The use of flash will give us the flexibility to make changes according to the need of the sponsors after the initial prototype is complete. Choosing the PIC24FJ128GA010 gives us the asset of having flash memory.

#### 2.3.1.4 Development tools

The development tool that was used by the previous group was recommended by the manufacture. The Explorer 16 Development board is manufactured by Microchip which is the same company as the PIC24FJ128GA010 used as our computation device. The starter kit includes an MPLAB ICD2 In-Circuit Debugger, the Explorer 16 Development Board, a 9V universal power supply, a serial cable, the MPLAB Integrated Development Environment (IDE), and the Student Edition of the MPLAB C30 C Compiler for Microchip‟s 16-bit devices, with tutorials and user manuals on CDROM. The Explorer 16 Development Board comes with an Alpha-numeric 16x2 LCD display, Microchip‟s TC1047A analog output temperature sensor, and the PICTail Plus connector for use with future expansion boards. The board is also designed to easily connect to a computer via USB or RS232 interface. Full documentation including User Guides, schematics and PCB layout is included on a CDROM.

Considering that Jerthwin is mainly doing all the embedded software for the PIC, the fact that the microcontroller is compiled in C is ideal. Due to the fact that we have only one computer engineer on the team, we needed the next strongest programmer to step up to do the embedded software. Therefore, the MPLAB C30 C compiler and the MPLAB Integrated Development Environment are ideal for our project because it allows us to program the microcontroller in the C language. The C language is a high level language which enables us to easily program the microcontroller, as oppose to assembly. The C language is very familiar to all of us studying engineering which makes it easy for everyone to participate in the integration and testing of the code. The Explorer 16 Development Board is compatible with many daughter boards that can be connected to the Explorer 16 to program desired chip. The previous team purchased daughter boards from Microchip which we will be using.

#### 2.3.1.5 Overall architecture of system and Cost

The components that were used by the previous group and will be used by us, uses a modified Harvard architecture, and therefore the microcontroller should use it likewise. The Harvard architecture describes a system with various paths for instructions and data. The majority of the processors that the previous team researched that met the specs from the sponsor used either Harvard architecture or Modified Harvard architecture.

The PIC microcontroller is fairly cheap and shouldn’t much of a disturbance to our budget of $1500. The PIC is only a small percentage of the total cost of the system although it is important in the system. The development board is much more expensive than the microcontroller itself, but because we are using multiple parts from microchip, we are able to use the same development board to program multiple devices.

#### 2.3.1.6 Voltage and current requirements

The voltage that is driving our main control unit is assumed to be 24 V RMS AC. This is determined due to the fact that the system will be installed where there was previously a thermostat mounted or installed in a new HVAC system as the main thermostat. It is of industry standard to build houses that supply the 24 V(RMS) for the HVAC system. The PIC needs between 3 V and 3.6 V to operate which would require use to voltage regulators, otherwise, the PIC will be damaged. Of course, before we feed the voltage regulators, we need to rectifier the 24 V(RMS). All of this will be taken care of in the PCB Design section of the paper. To iterate, our team has chosen the PIC24FJ128GA010 as it is High Performance, 16-Bit Digital Signal Controller manufactured by Microchip.

The PIC24FJ128GA010 has operation range, CPU, I/O, Communication, Analog to Digital Converter, and other characteristics that are essential to our design.

The operating range of this microcontroller is typical of industry standards which is usually from 3 -3.6 Volts. The typical operating voltage is 3.3 VDC. When the microcontroller is powered by a voltage between these parameters and kept between the temperatures of 40°C and 85°C, it is capable of up to 40 MIPS (Millions of Instructions Per Second).

The microcontroller is made to be programmed in the C language which is the most standard language of all. This is ideal to have a microcontroller that is programmable in C as we all in the group have experience in C. The microcontroller contains 16-bit wide data path which will be sufficient in accuracy to accomplish our task. There are two 40-bit accumulators to store our values of temperature and relative humidity (values should never exceed 14 bits), 32/16 and 16/16 divide operations, and 16 x 16 fractional/integer multiply operations to convert the temperature and relative humidity values from the form they are outputted by the sensor to an understandable form so that the user will be able to comprehend. This microcontroller has 256 Kbytes Flash memory. The 30 Kbytes RAM on the chip should allow the chip to process the data in a timely manner and then communicate it to the ARM processor. The CPU characteristics are presented non-sequentially below.

* C compiler optimized instruction set
* Modified Harvard architecture
* 16-bit wide data path
* 24-bit wide instructions
* Linear program memory addressing up to 4M instruction words
* Linear data memory addressing up to 64 Kbytes
* Two 40-bit accumulators
* Sixteen 16-bit General Purpose Registers
* Software Stack
* 32/16 and 16/16 divide operations
* 16 x 16 fractional/integer multiply operations
* 256 Kbytes Flash Memory
* 30 Kbytes RAM

The microcontroller has 85 I/O ports which can be programmed to other devices. This gives us sufficient room for upgrading the system based on the sponsor’s request. System expansion can be done once we have completed the initial prototype. The I/O pins can output between 3.0 and 3.6 V and 4 mA. The input and output significant specs are listed below.

* 85 programmable digital I/O pins
* Output pins can drive from 3.0 – 3.6V
* Wake-up/Interrupt-on-Change on up to 24 pins
* 4 mA sink on all I/O pins
* All digital Input pins are 5V tolerant

The SPI will be used to interface the PIC with the XBEE wireless chip which allows the secondary microcontroller to communicate to the microcontroller on the main control unit. The microcontroller allows for 2 modules to be connected via SPI at one time which then leaves us with space for one module if we chose to add to the system.

* 2 modules supported
* 8-bit and 16-bit data supported
* All serial clock formats and sampling modes supported

The I2C protocol will be used to interface the PIC to the temperature and relative humidity sensor and also the Carbon dioxide sensor. The microcontroller can support up to two modules at one time which leaves none else for us to use. I2C nuances relevant to our project is as below.

* 2 modules supported
* Interrupt on address bit detect
* Interrupt on UART error
* Wake-up on start bit from sleep mode”

The microcontroller has two ADC modules which will allow it to convert the analog signal (voltage or current) to a digitalize form. This feature will be important to us because we are connecting sensors to the microcontroller and that only way for the microcontroller to be able to communicate with the ARM and the secondary PIC is by converting that voltage into a high or low state. Without this feature, communication would be a more difficult task.

* 2 ADC modules
* 32 channel

This PIC is going to be used as an information source to the ARM processor where all the processing is done. We have sufficient memory to put in the programs that are going to be needed to massage the data from the sensors and send data to the relays. With the 30 Kb RAM, we feel like we have enough to have a well pace system.

We really selected the PIC to be just the information processing unit simply because we were convinced that the ARM would give us great flexibility in accessing the internet and having a very responsive and creative User interface. The decision was made also because we felt the previous team did a good job in selecting a microcontroller that gave us the flexibility to add features to the system. It was to our advantage to stick with the PIC because it would put us in a position where we wouldn’t have to reinvent the wheel. We are hoping to use some of the code from the previous and give the most important upgrade tasks to the new ARM processor which we feel is more robust and exceptional.

During development, we must stay within the specified absolute maximum electrical values. We must avoid subjecting the chip to extreme temperatures, input voltages it is not designed to handle, drawing or sending too much voltage or current from the I/O pins. The parameters are:

* Ambient temperature under bias: -40°C to +125°C
* Storage temperature: -65˚C to +150˚C
* Voltage on VDD with respect to VSS: -0.3V to +4.0V
* Voltage on any pin that is not 5V tolerant with respect to VSS: -0.3V (VDD + 0.3V)
* Voltage on any 5V tolerant pin with respect to VSS when VDD ≥ 3.0V: -0.3V to +5.6V
* Voltage on any 5V tolerant pin with respect to VSS when VDD < 3.0V: -0.3V to (VDD + 0.3V)
* Voltage on VCAP/VDDCORE with respect to VSS: 2.25V to 2.75V
* Maximum current out of VSS pin: 300mA
* Maximum current into VDD pin: 250mA
* Maximum output current sunk by any I/O pin: 4mA
* Maximum output current sourced by any I/O pin: 4mA
* Maximum current sunk by all ports: 200mA
* Maximum current sourced by all ports: 200mA

N.B Exposure to maximum rating conditions for extended periods of time may affect device reliability.

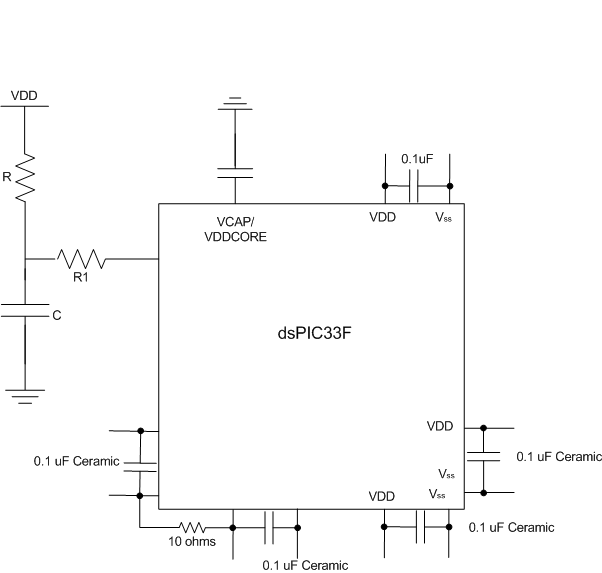
2.3.1.7 Pins

The PIC24FJ128GA010 microcontroller has a total of 100 pins on the chip. Some pins are required to be connected at all times, while some will be connected depending on our final design. The importance of knowing how and which pins to connect are vital to implementation of this device. Further the type of connection will be very important in determining which pin the line is connected to. Some pins will be left unconnected or floating. Pins that are required to be connected at all times are shown in Table 3. The Pin Name, Pin Type and Description are included in the table.

|  |  |  |
| --- | --- | --- |
| **Pin Name** | **Pin Type** | **Pin Description** |
| SCK1 | Input/Output | Synchronous serial clock input/output for SPI1 |
| SD1 | Input | SP1 data in |
| SD01 | Output | SPI1 data out |
| SS1 | Input/Output | SPI slave synchronization or frame pulse I/O |
| SCK2 | Input/Output | Synchronous serial clock input/output for SPI2 |
| SDI2 | Input | SPI2 data in |
| SD02 | Output | SPI2 data out |
| SS2 | Input/Output | SPI2 salve synchronization or frame pulse I/O |
| SCL1 | Input/Output | Synchronous serial clock input/output for I2C1 |
| SDA1 | Input/Output | Synchronous serial data input/output for I2C1 |
| U1CTS | Input | UART1 ready to send |
| U1RTS | Output | UART ready to send |
| U1RX | Input | UART1 receive |
| U1TX | Output | UART1 transmit |

**Table 3 PIC pin layout**

It is also required to use decoupling capacitors on every pair of power supply pins. Our printed circuit board design will include these capacitors as well as the resistors pictured. These capacitors will be considered to be part of the PIC microcontroller and will be thought of as connected to the PIC from here on when the PIC is referenced pictorially from this point forward in the paper. The PCB designs will include the capacitors, as they are necessary for the proper function of the PIC and will be considered a given for the devices full functionality. A diagram is shown below in Figure 9.



**Figure 9 Minimum Connections**

The PIC shares the main control unit with two devices requiring SPI connections, two devices requiring I2C connection and one device a UART connection. Table 3 shows the pin name, type and description that will be utilized to accomplish this interfacing.

### 2.3.2 ATMEL

Atmel offers both microcontrollers based on the ARM Architecture – based on the Cortex line – and Harvard Architecture under its AVR line which is also the same Architecture that our existing Microchip PIC microcontroller(s) use. For comparison of the AVR line, being the same processor architecture, the differences between the offerings are not significantly different for the needs of our project outside of the need for support for external peripheral boards like Zigbee/XBee, which they both have. Where we feel the products are significantly different for our project is the Integrated Development Environment (IDE), and product technical support.

Comparison was done by installing both development tools from the website. While we are also fortunate enough to have a development board for the PIC microcontroller, we don’t have a development board for the Atmel. As such, we will simply compare features of the IDE’s provided without any interaction with hardware. Fortunately Atmel offers a new version of its IDE as of approximately a month ago – AVR Studio 5.0. The features of the IDE seem to turn the coding environment into a modern compiler experience, which the MPLAB IDE v8.46 by Microchip lacks. While it is still in beta, by the time we use the development environment for our coding the product should be more mature and concerns of bugs should be mostly alleviated.Being beta software there however could be a lot of bugs and compatibility problems. We decided that another good way of measuring how good of a platform is it was is via support forums. What we found was discouraging. The forums were full of complaints. One in particular was over 10 pages long with, at the time of this writing, a vast majority complaining about issues. In light of this information, we decided that the older software be a better use from a design standpoint. We didn’t want our time spent beta testing software and discovering bugs for Atmel. The features of the older software were not significant different over Microchip’s and taking this into consideration as well as our limited experience with both the software development environments we felt that there was not a significant difference in the development environment between the two processors. Our comparisons were then based on the technical support both by the companies and 3rd party forums, as well as documentation provided.

Atmel has an amazing amount of community support. There exists an entire 3rd party forum specifically dedicated to Atmel products, AVRfreaks.com. The forums are extremely active and have some 400,000 posts as well plenty of examples of informed users willing to help each other. Along with their official forums in which we found to have many answers to questions resolved quickly and satisfactory we feel extremely comfortable choosing them should we need support or technical assistance. They also have a lot of third party device support by the listing on their website. We found microchip to also have a decent level of support although not as significant as Atmel, at least in terms of forum activity. Microchip has several forums divided more into specific areas which seem to lead to easy to find questions that have been answer before and for that specific topic. While they weren’t as active as Atmel by our comparison, they were still plenty of activity and good answers to many of the questions posted which we felt showed that the product was well supported. The seeming lack of activity could be due to the division of the support forums. Overall we feel there was no clear winner here, and we felt we had to side with the PIC microcontroller here as the previous group has already invested in the platform. Had AVR Studio 5.0 been a more mature product, Atmel would’ve easily been our choice for a microcontroller.

## 2.4 Secondary Microcontroller/ Remote Sensor:

The secondary unit which reads the temperature and humidity outside of the building needs a low power microcontroller since it will be powered by a portable power source. This power source could be either a battery or a solar power supply. This PIC will interface with the sensors and 802.15 XBee wireless chip. We have chosen to stick with the microcontroller chosen by the previous group PIC24F04KA201. This chip has 20 general purpose 16 bit flash microcontroller manufactured.

The series 1 secondary controller used a PIC24F04KA201 chip to control the outdoor temperature sensor. It connected to the main controller through an XBee chip along 802.15 IEEE protocol and powered itself using 2 AA batteries. The original printed circuit board contained several shorts. The original board will either be completely fixed and the new parts added or a new solution is to be found.

### 2.4.1 General Description

The secondary microcontroller/remote sensor is to be placed outside and send the outside temperature and humidity data to the main control unit to decide the most economical way to cool/heat the house depending on the desired temperature inside in relation to the outside temperature and humidity. The original device was planned to use a zigbee chip, which was soon phased out in favor of the more economical Xbee chip and it's protocol. Power was supplied to the system using batteries. The device was supposed to have a long sleep cycle in order to save battery life. The new device should be able to easily do all the above, and proved an option for wired connectivity (preferably through a twisted pair wire set) that could also provide power over ethernet. Solar power should also be at least considered for this outside unit.

The information for this microcontroller will be displayed in the following paragraphs of this section which includes the operation range, central processing unit, I/O, Analog to Digital Converter and other general characteristics.

#### 2.4.1.1 Operating Range

The operation range of this microcontroller is to industry standard. Since this microcontroller requires a supply voltage, it has to be low powered. The microcontroller is going to be attached with the XBee devices and with the temperature and humidity sensor. These components themselves have to be low powered. The whole secondary unit will be battery powered and having the low power supply voltage is essential to our design specifications and to keep with the temperatures of -40°C and 85°C. The PIC24F04KA201 features new nanoWatt XLP (extreme Low Power) technology for the low power consumption that we desire.

1. Supply voltage 1.8 V – 3.6 V
2. nanoWatt XLP
3. -40°C - 85°C

The microcontroller contains power saving technology such as On-the-Fly Clock switching, Doze mode and Instruction-based power saving modes. The PIC24F devices have two special power saving modes, entered through the execution of a special PWRSAV instruction. The sleep mode halts all code execution; idle mode halts the CPU and code execution, but allows the peripheral modules to continue operation, such as the XBee and temperature/humidity sensor. The deep sleep mode stops clock operation, code execution and peripherals. It also freezes the I/O states and removes power to the SRAM Flash memory. A combination of all these methods will allow us to be effective in our power consumption.

#### 2.4.1.2 CPU Characteristics

1. Up to 16 MIPS
2. C compiler optimized instruction set
3. Modified Harvard architecture
4. 16-bit wide data path
5. 4 Kbytes Flash Program Memory for storing and executing application code
6. 512 Bytes RAM data memory

The microcontroller is capable of up to 16 MIPS (Millions of Instructions Per Second), which is the industry standard for a microcontroller of that size and power. We will be able to program the microcontroller in C which is ideal for our group since we are all acquainted with the C language. It will be responsible for taking in and distributing the data taken from the SHT21 sensor. The 16-bit wide data path will be sufficient for the processing of the temperature and humidity readings keeping enough accuracy. The processed information will then be passed along to the pic on the main control unit via the 802.15 XBee chip.

The internal flash program memory for any device is used to store and execute application code written by the designer. The 4 Kbytes of Flash memory will hold the programming code and allow the program to be modified by future teams. The memory is readable, writeable and erasable when operating at a threshold VDD of 1.8 V. 512 Bytes of RAM will allow us to run the low data RF module quickly and effectively without much delay.

This microcontroller also has 18 I/O pins that can be programmed to work with all the other devices that we are going to be using such as the sensors and XBee RF transceiver. Although this many input/output ports far surpasses our need, it is necessary as there will be future updates made to the system which would require more ports to connect more devices. It also has ADC modules to transfer input analog voltages into a digital number proportional to the input voltage and current.

#### 2.4.1.3 I/O Characteristics

The SPI module protocol will be used to interface the PIC with the Xbee wireless chip. We will use the I2C module protocol to interface the temperature and relative humidity with the microcontroller. This I2C module supports both master and slave modes of operation, useful for the sensor connection. The PIC has 18 I/O pins, a build in 10 bit A/D converter with up to 9 input channels. We will interface using three serial communication peripherals: SPI, I2C, and the UART.

#### 2.4.1.4 Cost of the original unit

While the cost of the unit is not the basis of the entire judgment call of what to do, it should be significantly noted. The cost of the first secondary unit should be similar to the cost of us building on. We will be adding one additional connection if it is kept and changing the powering of the system. The individual parts and cost that were used to manufacture the original secondary control unit were compiles and listed as well as totaled in Table 4.

| Product | Part Number | Manufacturer | Quantity | Unit Price | Subtotal |
| --- | --- | --- | --- | --- | --- |
| 5VDCtoDC  2xAA Battery | vpack5aa2 | bodhilabs.com | 1 | $10.95 | $10.95 |
| JST Vertical Connector | PRT-08613 | SparkFun | 1 | $.95 | $0.95 |
| JST to Female Connector | PRT-08599 | SparkFun | 1 | $1.50 | $1.50 |
| Voltage Regulator – 3.3V | COM-00526 | SparkFun | 1 | $1.95 | $1.95 |
| XBee 1mW Chip Antenna | WRL-08664 | SparkFun | 1 | $22.95 | $45.90 |
| 2mm 10 pin XBee Socket | PRT-08272 | SparkFun | 2 | $1.00 | $2.00 |
| RJ11 6Pin Conn. | PRT-00132 | SparkFun | 1 | $1.25 | $1.25 |
| Break Away Headers-Straight | PRT-00116 | SparkFun | 1 | $2.50 | $2.50 |
| Break Away Female Headers | PRT-00115 | SparkFun | 1 | $1.50 | $1.50 |
| Jumper - 2 Pin | PRT-09044 | SparkFun | 1 | $0.35 | $0.35 |
| Chip Resistor (10K Ω) | CRT0603- | Digikey | 2 | $0.16 | $0.31 |
| Ceramic Chip Capactior (10μF) | 587-2562-1-ND | Digikey | 1 | $0.54 | $0.54 |
| Ceramic Chip Capacitors (.1μF) | 490-1575-1-ND | Digikey | 3 | $0.02 | $0.05 |
| Pulse Proof, High Power Chip Resistors (470Ω) | 541-47OSADKR-ND | Digikey | 1 | $0.88 | $0.88 |
| PIC24F04KA201 (surface mount) | PIC24F04KA201 | Microchip Direct | 1 | $1.84 | $1.84 |
| Small PCB fab | (Eagle CAD Files) | 4PCB.com | 1 | $33.00 | $33.00 |
| SHT-21 Humidty and Temperature | Liquidware.com | Liquidware.com | 1 | $39.93 | $39.93 |
| TOTAL |  |  |  |  | $145.40 |

**Table 4 Secondary Unit Cost**

Texas Instruments makes EZ430-RF2500-SEH which is a wireless board with temperature reading capabilities and a solar panel. This device contains 18 analog and communication input/output pins to install a humidity sensor. The cost of this unit is $149.00, if the cost of a made remote PCB with solar panel exceeds this, a decision should be made.

### 2.4.2 Xbee vs Zigbee Wireless

There is a lot of confusion as in to the difference in Zigbee and Xbee. They are essentially the same thing, low power RF communications. Zigbee is a name brand that uses the Xbee style of RF transmission and adds a lot of end user functionality such as power monitoring. For the system we're using, it is accepted that Zigbee is extremely overkill for our system and the Xbee standard will suffice for our remote communication needs.

### 2.4.3 Wired Connection

While the main source of connectivity between the remote sensor unit and the main unit is via wireless, an option needs to be included for a possible wired connection. This needs to be done so that buildings with extremely thick walls can still have remote access, as well as the possible ability of power over ethernet.

An RJ-45 8 pin connector is available from Sparkfun for $1.50. This could either be added to the existing remote unit via the I/O lines or the TI Solar unit via its remaining I/O lines. Information can travel across the ethernet’s twisted pair lines and into the main unit via an installed connection there. The twisted pair CAT 5 over an Ethernet RJ45 connection is the most common us for data transmission in today’s computers. The twisted pair reduces or electromagnetic interference. Two inverted signals will be transmitted across two pairs of wires. Two sets of twisted pair cable will be devoted to data transmission. Each set will have one line dedicated to the actual data, and the other will have the inverted signal which will help eliminate unwanted noise, and allow the fast, reliable, and long distance data transfer. Data transmission can be as fast as 100Mbps with great accuracy of large distances.

The use of the RJ-45 was also chosen over the existing RJ-11, another twisted pair technology, because **IEEE 802.3at-2009** allows a power of ethernet up to 25 Watts. This technology could be used to power either system remotely, further making the need of batteries, and the sleeping code, less necessary.

If going with the main MicroChip PIC units, SparkFun makes PRT-08556 $23.95 to add power over ethernet and the DEV-08557 $69.95 That can decipher the signal from the power, power a pic processor, and allow internet connectivity. TI also offers a wide variety of power over ethernet options, some of which are much cheaper.

## 2.5 Wireless and Internet Connectivity

### 2.5.1 Overview – Wireless and Internet Connectivity

Our design decision to go with the ARM microprocessor also has the added benefit of allowing us greater flexibility with internet connectivity both wired and wireless, but especially wireless. Being able to run a \*nix based operating system will mean that unlike the PIC microprocessor, the TCP/IP stack will already be implemented in a mature operating system. Wireless card support with the processor and operating system would not have to be implemented. Diagnosing issues on the local network the device is deployed on will be significantly improved given the tools included in any modern operating system, but especially in the Linux operating system. Finally we have greater flexibility with how we plan on implementing remote access.

### 2.5.2 TCP/IP

The TCP/IP protocol, shortened for the Internet Protocol suite, is a non-proprietary communications protocol standard and is the basis for commutations on the internet. The protocol defines a way of formatting and encapsulating data so other devices a network can understand the data and communicate. Figure 10 shows the way data is encapsulated in TCP/IP as it moves out to be broadcast on the network to other devices.

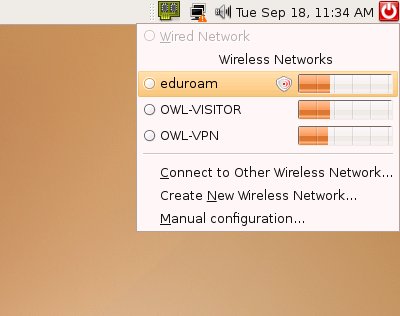


**Figure 10 Encapsulation of data in TCP/IP**

To conceptualize the way the protocol does this, it’s divided into so four called abstraction layers, the Application layer, the Transport layer, the Internet layer and the Link layer. By using the ARM microprocessor the implementation of the TCP/IP stack has been significantly simplified by using a sufficiently mature operating system like Linux on the ARM microprocessor. The operating system implementing the stack allows us to ignore the more complicated implementation of the TCP/IP stack via C code provided by Microchip for the PIC processor. This also allows us to not have to re-invent the basic functionality of wireless internet. Linux provides a mature TCP/IP stack and wireless internet that ensures that setting up network communication via TCP/IP will be aside from basic configuration issues with home network routers like NAT address translation, and dealing with possible firewall issues for remote access via the internet we will almost entirely have to work with the higher level application layer.

### 2.5.3 Operating System - Wireless and Device Driver Implementation Maturity

Probably the most significant change from the way internet access would have been implemented with a PIC microprocessor is that we no longer have the added difficulty of deciding on a wireless card to communicate with the board and work with the TCP/IP stack implemented in the PIC. Configuring was complicated and had to be done in the source code. This was unfeasible for an installation stand point as we would have to have known home wireless network information where the device were to be installed beforehand or implement a complete user interface that allowed for changing of such features. Knowing the wireless information like the wireless password, frequency, and other information about the network would’ve been a hindrance to deployment. The alternative, developing a complete GUI just to be able to configure the device properly as it would be in field would’ve diverted resources from other more important tasks. Furthermore implementing TCP/IP seemed to be a significant drain of resources of the previous team and it seems to be a tricky thing to get done right. The ARM board we choose will likely have both integrated wired and wireless communications. If it doesn’t the driver support of Linux operating systems means that so long as we choose a Linux operating system and a device with Linux support it should be significantly easier to add such a device compared to the PIC. We also are able to use what many Linux distributions have with the ability to scan for wireless networks, enter in a password, and the ability to easily interact and diagnosis networking problems on the fly. Figure 16 shows the GUI wireless network selection in the Linux distribution Ubuntu – a likely choice for our ARM based microprocessor – and Figure 11 shows the configuration dialog.



**Figure 11 Wireless Network Selection in Ubuntu**

The PIC microprocessor used instead its own propriety code and the wireless network name, password, and encryption would have all had to be known before deploying the device on the site, put into the programming code of the PIC microprocessor, and then programmed on the device. GUI implementation would’ve been very difficult given time considerations and so manual pre-deployment configuration would’ve be the most likely route of configuring especially if there were issues automatically configuring it, which is common in the wireless networking world even though the technology is significantly improved on mature platforms. The PIC code seemed to be archaic in this sense that to interact without modifying the code we’d have to implement our own interface – essentially redoing what has caused many distributions of Linux issues in implementing right in the earlier years of wireless networking. This would’ve been a waste of our time and resources implementing, and taken away from other more important product features like scheduling. While Linux can still be tricky with driver support even for an experienced Linux user, we feel the change offers a significantly decrease of potential problems. The chances that our time will be spent working on the more advanced functionality of the device rather than programming the basic processor to wireless card communication and implementing a GUI that works is significantly improved. Figure 12 below shows the WPA login screen.



**Figure 12 Wireless Security Configuration in Ubuntu**

### 2.5.4 Server

There are many ways to implement the ability to remotely access information on the HVAC system and control the system. One of these ways is with a Hypertext Transfer Protocol (HTTP) server. We plan on having an Application Programming Interface (API) that will be created on the PIC that allows us a standard way to interact with an application running on the PIC to request information like temperature and humidity from it via I2C or similar serial interface. We can simply store these values to a data file or in memory on the ARM processor and have use the data as variables on a webpage which will be accessible via the HTTP server. There are many options for implementing an HTTP server in Linux. One common HTTP server implemented on Linux is the open source Apache web server. Apache is widely implemented with over 59.11% of all websites on the web using Apache and of the million most busiest Apache is run on nearly 67% of them as of February 2009. The program is also well documented probably due to the nature of being open source software where anyone can contribute code the website documentation is also setup in a similar fashion with Wiki knowledge bases – or user created guides on the software. On a lower end system Apache may cause performance issues if implemented however due to how powerful the software is, it may take more memory and resources for an embedded system than one made specifically for embedded systems. This could be an issue if we choose a board with lower RAM, CPU, and ROM specifications which we might do due to monetary and availability reasons. Many of the boards we have looked at should not be an issue in this regard, however the possibility exists that this may be too resource intensive. We could also implement a basic application coded in C, C++, or Java programming languages that would act as a sort of API much in the same way the PIC does only via TCP/IP over the internet. Java lends well to this as it is a programming language designed to very easily interface with TCP/IP and network devices, unlike C or C++. Java however is a resource intensive programming language. Being developed as a cross platform programming language, Java code executes on a so called *virtual machine*. In the case of the Java programming language a virtual machine is a set of software made specifically for a particular operating system in which the Java code executes. The virtual machine then translates the Java programming into *native code* or code that works on a more basic level with the particular operating system. The advantage of this is that the same code can run on any operating system that has a virtual machine made for it. The downside however is that due to the code having to be executed and translated in the virtual machine before it then gets executed on the operating system the program is more resource intensive than simply natively compiled and executed code.

On an embedded system with limited resources this could make for a less responsive user experience and possibly complicate meeting other design specifications. An alternative to both of these implementations is an open source HTTP server for Unix based operating systems called THTTPD (tiny/turbo/throttling HTTP server). THTTPD is designed for deployment in systems where resources are limited as it is designed for speed and a little memory usage. The entire executable code has a memory size of about 50 kilobytes. While there are many other HTTP servers suited for resource limited embedded system applications, THTTPD seems to have the added benefit of having many large popular websites use the software. Furthermore THTTPD is also included as an easily downloadable package within Ubuntu software repositories. While Apache also shares this compatibility and support of Ubuntu, THTTPD does so and is focused specifically for being minimal in use of system resources. Should implementation of Apache be too complicated or resource intensive, THTTPD may be a good alternative choice in either of these cases.

### 2.5.5 Web Application

Creating a web application that can be interacted with on a variety of platforms - such as Windows, Linux, Android, and IOS - is very appealing. Such an implementation could easily be done with a webpage coded using any combination of the following web script programming languages: HTML5, Javascript, and PHP. So long as the page is coded to in line with web design programming standards, many modern browsers such as the open source Webkit page rendering engine and the open source Mozilla Firefox web browser. Web kit is implemented in Google’s cross platform Chrome Web Browser – available for Windows, OSX and Linux, Google’s Android default web browser, Apple’s Iphone IOS only web browser, and finally in Apple’s Safari web browser for OSX. Mozilla’s Firefox is used by approximately 30% of the world web users and is also web standards compliant as well as cross platform. Both Chrome and Mozilla Firefox are cross platform for Windows, OSX, and Linux. Targeting both of these browsers should be easy as they both follow the same open web standards and will allow for the remote access and control of the HVAC system on a wide variety of platforms with a standardized interface.

Scripting languages are what many if not most web applications are programmed with. As mentioned above HTML, Javascript, and PHP are some of these languages. We will expand on these briefly as well as introduce a couple of new scripting languages we may implement. HTML, or Hypertext Markup Language is the most common and language for webpages. The standards of which are defined by the World Wide Web Consortium (W3C). At its very basis the language standard defines a set of structural semantics for images, text, hyperlinks, forms and other interactive and non-interactive page elements. In implementing a web application HTML is unavoidable and will definitely be used as a scripting language should we choose to implement it this way.

While not as fundamental to webpages as HTML, Javascript is increasingly becoming prevalent on websites. It is a form of EMCAScript which is a scripting language standardized by Ecma International. It is important to note that so there is no confusion between Javascript and the Java programming language, Javascript is not Java. This is a frequent misconception. Javascript is supported by all major web browsers. A feature of Javascript is that its code is run locally on a user’s web browser which allows for a native application like response. It is also capable of detecting user interaction which more feature rich modern websites like Google’s Gmail combine with other languages to create a truly interactive website that functions very similarly to a native application. Javascript is part of the AJAX – short for Asynchronous JavaScript and XML – group of interrelated web scripting language and development methods. It is a fundamental scripting language to the web to move more toward a platform for applications. AJAX and possibly Javascript seem outside of the initial implementation of a web application for our purposes as they are unnecessary to achieve our initial design goals however may have a place in a future more feature rich version. PHP is a scripting language that can be used for many purposes although its original purpose was for development of dynamic web pages. To implement PHP, it must be embedded into the HTML code and that HTML code must be hosted and executed on a web server which has a PHP code execution module. This module then produces the web page content. PHP is highly deployed with approximately 75% of all webservers using it as their server side programming language. For creating a more advanced website that has dynamic content changes like temperate and other information remotely, PHP might be a good solution. Apache, an HTTP server we are considering deploying for our design, has support for PHP via a module. The PHP module is the most popular of all the Apache HTTP server modules.

### 2.5.6 Client Side Application

Only in the last couple of years has the idea of a cross platform web application been a viable reality. Traditionally implementation of interacting with a remote service was done with a client side application. Often times the application was written in C or C++ for one operating system and then ported to other operating systems, a time consuming process. The Java programming language allows for easy implementation cross platform and is a likely candidate for implementation of our client side application were we to forgo using other viable methods of implementing remote access features. The program would connect via TCP/IP to a basic server on the ARM microprocessor and then display that data and allow for the client to change settings on the device. One down fall of java implementation would be that IOS used on the Iphone would not be supported as applications developed for IOS must be written in Objective C as per Apple’s developer agreement. A Java program in this implementation suffers from the same drawbacks as the server application in that Java can be a resource intensive programming language. For an embedded system with possibly limited resources, especially as the unit is scaled to manufacturability, Java may not be a good design choice.

### 2.5.7 RSS Weather Feeds

A design specification for our project is the ability to display local weather information from the internet. We decided the best way to implement this would be with Really Simple Syndication (RSS) feeds for such data. RSS is a type of Extensible Markup Language (XML) and follows a similar formatting of the data as XML. Many weather sites, such as Accuweather.com, Weather.com, RssWeather.com, and the National Oceanic and Atmospheric Administration’s (NOAA) own weather.gov all offer RSS feeds for current conditions as well as forecasts for many locales. Writing an application on the ARM microprocessor to access this information once a source feed is chosen should be relatively easy. The RSS feed offers a standardized schema with tags and attributes which allows for us a way of locating and reading the data within the data format. In choosing an RSS feed we will have to take into consideration the source of the data’s right to reuse the information. One potential source of RSS feeds is rssweather.com. RSSweather.com’s RSS feeds are specifically stated as public domain at the bottom of their website as they pull their data from NOAA and have forecasts included in the RSS feed in what is an informative but concise manner which allows us to display it properly on a screen of our size. It is not clear however whether or not we can re-use their information simply because their data was constructed from public domain sources. As such we will stick with the NOAA feeds.

Breakdown of RSS Feeds

RSS is a specific type of Extensible Markup Language (XML) standard which is defined by the World Wide Web Consortium (W3C). The W3C is a non-propriety standards organization. The W3C is most famous for being the organization that standardizes versions of hypertext markup language (HTML). XML and by extension RSS both are very similar to HTML due to their use for web applications and their origin in the same standards organization. An example of an RSS feed can be seen below in Figure 13.

|  |
| --- |
| **<?xml** version="1.0" encoding="UTF-8" **?>**  **<rss** version="2.0"**>**  **<channel>**  **<title>**RSS Title**</title>**  **<description>**This is an example of an RSS feed**</description>**  **<link>**http://www.someexamplerssdomain.com/main.html**</link>**  **<lastBuildDate>**Mon, 06 Sep 2010 00:01:00 +0000 **</lastBuildDate>**  **<pubDate>**Mon, 06 Sep 2009 16:45:00 +0000 **</pubDate>**    **<item>**  **<title>**Example entry**</title>**  **<description>**Here is some text containing an interesting description.**</description>**  **<link>**http://www.wikipedia.org/**</link>**  **<guid>**unique string per item**</guid>**  **<pubDate>**Mon, 06 Sep 2009 16:45:00 +0000 **</pubDate>**  **</item>**  **</channel>**  **</rss>** |

**Figure 13 RSS feed example code**

(Use and adaption is permissible for fair use under the Creative Commons License)

The XML declaration is placed at the start of the document that can tell a program reading it information what type of document it is. Since RSS is a type of XML, the header tells the program that this is an XML document rather than an RSS feed. In Figure 18 that is the very first line. The next line is the RSS version which is also important as this declares the version of RSS. In this particular example the version is 2.0. Finally for declaring the start of data, <channel> is used, those familiar with HTML will see that it’s similar to HTML’s <body> markup. Within the channel markup are three required elements. These are the <title>, <description>, and <link> elements. The title gives a name to the channel. In our example it is “RSS Title” but for a weather feed this will most likely be “Forecast for Sunday” or something similar. The description element gives a body of text to explain the title. In our example it is “Here is some text containing an interesting description.” but for a weather feed this will most likely be “Mostly Sunny with a high of 90. Lows around 60. Winds are 5 MPH to the east.” or something similar. The link element is the URL that points to the website of the data channel. In our example it is “http://www.wikipedia.org” but for a weather feed it will most likely be “http://www.weather.gov”. Finally there are other optional elements like <pubDate> which contains information about the time the information was updated.

Our program will essentially read the XML RSS feed documents looking for these elements. Once found the program will then take the data from that element and store it in a section of the program designed so it will be displayed to the user in the GUI. The section the program stores it in will depend upon the element. We then would display this data to the user when the “local weather” option on the Graphical User Interface (GUI) is selected. A mockup of such a design is shown in the software design section of this paper.

## 2.6 Power System

The main power supplied to HVAC system will be supplied through a 24VAC that is typical of most house AC control units. HVAC units run on 220V, but are controlled through a series of 24VAC relays. Most other systems made are generally made with a typical 115VAC planned input, so the transformer and DC conversion is as simple as plugging in the accompanying cord and transformer. We don't have this luxury as our system needs to be powered by the 24VAC. The relays will be controlled by a micro-controller as opposed to the typical linear control system seen prevalent in most traditional settings today. The series one HVAC system had several design issues with the power system that lead to short circuits, voltage regulators overheating, and the need for two external power sources for both the main control unit and the display system.

### 2.6.1 Main Unit

The design of the power system on the original HVAC system attempted the use of a full wave rectifier circuit connected to a voltage regulator. 24VAC is the RMS value meaning the peak value of 34V was rectified and sent right into a voltage regulator after being smoothed out by a capacitor. The voltage regulators (12V and 5V) in the device are pushed to its thermal limit when the input voltage going in is 30-32V. The possible solution is to step down the voltage. Also, instead of sending the not rectified voltage straight into the 3 regulators (one 12 V regulator and two 5 V regulator) perhaps taking the 12 V regulator and throwing that into the 5 V rectifier instead of straight from the full wave rectifier. This will limit the thermal stress introduced to the regulators, as well as help keep the unit slim due to less need of heat sinks. Adding heat sinks to the regulators need to also be researched, as well as introducing an already built 24VAC - 12VDC regulator.

#### 2.6.1.1 Rectifier

After doing design research on the original HVAC unit there were several flaws found in the power system. The full wave rectifier lacked an adequately sized capacitor. This made the rectified signal too bouncy. Further, the rectified signal was then sent parallel to the voltage regulators, one 12 volt and two 5 volt. One of the 5 volt regulators then fed to a 3.3Volt regulator. These all failed due to not having adequate current tolerances.

#### 2.6.1.2 Distribution

The new design will send the rectified signal to a significantly more tolerant 12 volt regulator that can handle the current to push the entire device. The voltage regulator originally chose in the first version was the MC78L12BP-AP from Digi-Key. Analysis of the data sheet shows that this part is a low current regulator that likely overheated due to too much current. I would suggest using the TORREX DC/DC step down, Newark part 59M3504. It has a max output current of 3A and can handle 18 Volts input while maintaining a steady 12V output at safe operating temperatures. This 12V output is enough to fully light up a 7 inch LCD touch screen and will feed the 5 volt and 3.3 volt regulators. It is important to our sponsors and us that we remove the need to have both units plugged in. The system needs to get all of its power from one source on the wall, the 24VAC and convert it to the usable form needed.

The 12 Volt regulator will then be split off to power the ARM board. The connection will be made simple as device installation needs to start taking some priorities. The 12 Volt regulator with then power the 5 volt regulator, also a little more tolerant will feed one 3.3volt regulator. The 5 volt regulator originally chosen in the project was Micro Commercial Co’s MC78L05BP-AP. This has a maximum output current of 40mA which may or may not have lead to this parts demise. It will be replaced with On Semiconductors part MC78L05BP-AP with an output current of 1A. While these numbers may be a bit extreme, this unit will be always on, and heat of circuitry will be an issue on parts too close to threshold. It is important that the powering of this unit be safe, as we are regulating down from a 24VAC to a stable 12.5, and 3.3V to power sensitive equipment. This will all be on the printed circuit board containing the main micro-controller. These controllers will easily be able to power the lower power needs of the PIC processor as well as the sensors and wireless connectivity.

#### 2.6.1.3 Vent Servos and Mood Scents

Distribution of a DC signal over the distances of a house or environment predicted. Because of this, we will just continue to use our 24VAC as our signal to turn on both the vent dampers and mood scents. This also works perfectly, as most pre-built electronically controlled dampers operate on a houses existing 24VAC system. Additional relays can be added and controlled by the microcontroller, which will easily turn on and off zones and scents. Eventually, relays should probably be built into a separate relay box, as the majority of the size of the main control unit will eventually be relays.

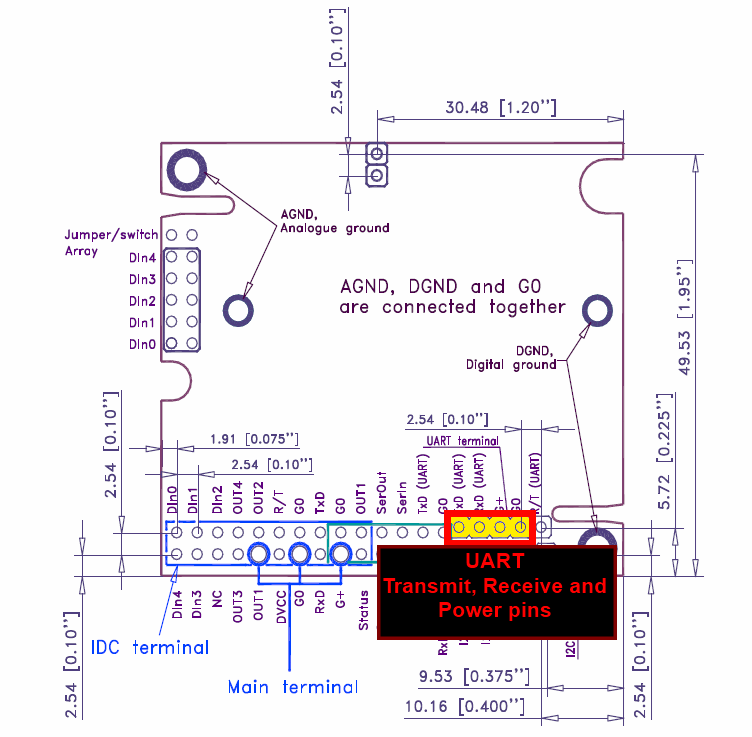
### 2.6.2 LCD Screen

Much of the power draw on this circuit will be due to the LCD touch screen device. The entire purpose of having a 12V regulator is to full light the LCD screen. In order to save power, and increase the life of the device, power saving needs to be investigated software wise and hardware. The ability to dim the screen with user input needs to be made an option. This, just as in your cell phone requires much less power and the less bright the screen is lit the longer its pixels life. Further, just like a computer screen, when the user walks away from the device there needs to be a time out to either blank the screen or screen saver. A screensaver option may be the best fit so as the touch screen interface requires the screen be actually on.

### 2.6.3 Remote Sensor Unit

Currently the remote unit is powered by two double A 1.5V batteries. It is the responsibility of the user to unplug the jumper in the unit or take out the batteries to save its batteries life. A sleep mode was initially installed in the internal software of the secondary unit’s controller, but had to be abandoned due to issues. Before investigating new ways to power the secondary unit, reducing the amount of power it consumes is most important. A sleep mode needs to be executed regardless of power system installed in order to help save the life of the system. We also researched the potential of adding a Senseair CO2 K33 sensor to it which incorporates both battery savings in being able to log data to a ROM on the device, and a temperature and humidity sensor on board, so we can save cost in going with going with a chip that has just two digital communication ports.

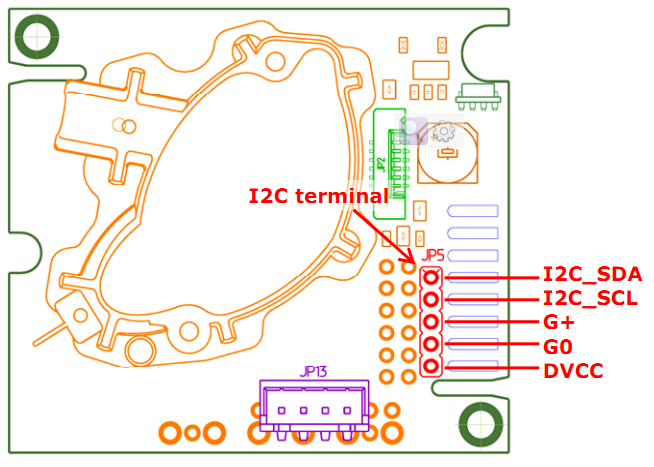
The Senseair sensor unit has some other important design considerations since it employs analog to digital circuitry onboard. While the data sheet said that we might run into issues with interference from Electromagnetic Fields (EMF’s) and should we need to we should ground (AGND) connections which are also the edge connectors as shown in the top left of the data sheet. Our testing showed that this wasn’t necessary with our digitial communication and thus wasn’t implemented in that way. However the device is sensitive to electrostatic discharge and at all times we handled it with a grounding cable to ensure we were properly grounded. We ordered two wrist grounding straps for this purpose and having them and using them at all times prevented us from wasting time with a damaged device. The connectors can be seen in Figure 15 below.



**Figure 14 K30 Schematic Highlighting UART**

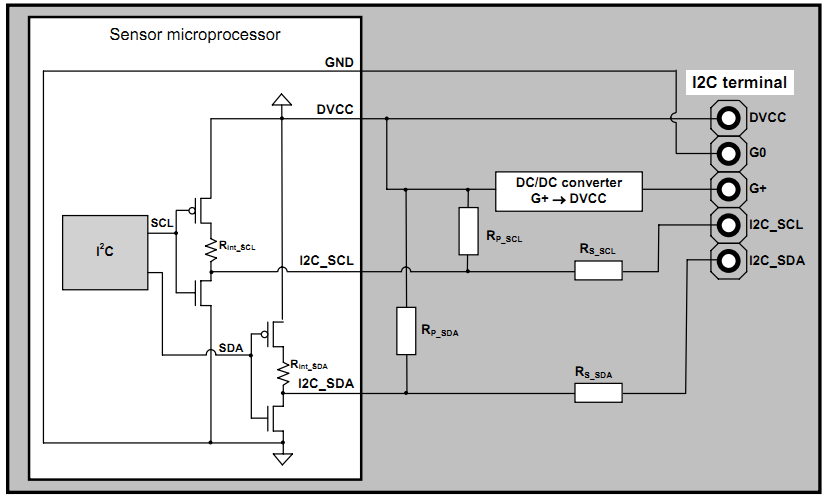
**(Reprinted with permission from Senseair)**

For the remote unit we decided to go with the Senseair K33 BLG/ELG module. The K33 chip gives us the flexibility we need for the remote unit in many significant ways. It has a significantly small current draw of ~240µA when configured to measure values approximately one per hour and absolute max voltage ratings of 4.75V to 12V which allows us to use many types of batteries. Here we opted and designed for 5V however we will implement a 9V design in the future to ensure we are not close to the minimum value, to overdesign and ensure we do not run into device issues due to having too little of a voltage applied. With that voltage range as well we of course still have the option of powering it as well via power over ethernet (PoE) or similar wired powering methods. The module is able to do this by having a built in nonvolatile random access memory (NVRAM) chip integrated to its design. This chip stores its readings into the memory and stores 5400 measurements including timestamps for all measurements. Also, the sensor isn’t just a CO2 sensor. It includes a built in temperature and humidity sensor by Sensirion, which is the same company that we choose for our temperature and humidity sensors. The chip schematic diagram follows below in Figure 15 showing the I2C connections only.



**Figure 15 Sensair K33 Schematic highlighting I2C connections.**

For battery operation the connections seen in the data sheets UART connection diagram, which shows the terminals on the opposite side of the chip. We decided against using the Vbat+ connection as we can use the G+ and G0 in its place because it offers circuit protection against reverse connections and current flows. As we can see in Figure 16 which is applicable to both the Senseair K30 and the K33 have a series of resistors that are internal with a series of external pull up or pull down resistors externally. The diagram is generic and has a note that the Senseair K30 model it is slightly different in that it has a 10R resistor between the output and the DVCC pin on I2C.



**Figure 16 I2C internal connections on the K30/K33**   
Note: There is an addition 10R resistor between  
output and the DVCC pin that is not shown  
(reprinted with permission from Senseair)

### 2.6.4 Dampers

Electronically controlled dampers are somewhat widely available online, such as Netropolus.com part 307107 for $59.97. They generally operate at 24VAC much like our AC and Fan units. Several of these could be purchased for the purpose of wiring in the home and controlled through the main micro-controller through relays exactly like was in the original unit to control fans and air conditioner. The power will the taken from the output of the wall before rectified and fed into a set of relays. In order to make the unit smaller, one day the relays should be separated from the main unit and put in an attic with just a wire controlling the relays and a simple smaller microcontroller turning them on or off. An NPN transistor will be placed between the microcontroller and the relay. This will have a twofold purpose. One will allow us to adjust the voltage to the correct 3.3V needed for the relay to be switched off. Secondly, an NPN transistor is essentially two diodes placed end to end. This diode between the relay and the logical silicon circuitry of the system will be necessary to prevent a bounce back spike from the relay switching caused by the internal magnetism inside of a typical relay. This bounce back can wear down and destroy a system over time, as it can provide short but intense amounts of increase current across the circuit. The diode will prevent this.

There are two types of electronic dampers, normally open and normally closed. Normally open dampers are open when off and when a voltage is applied they go off. Normally closed works in the opposite fashion, they are closed until a voltage is applied to open them. A combination of these dampers would probably be the best to apply to require the least amount of electrical work of the dampers. If it is determined that one location will have the damper closed more frequently, then a normally closed should be installed, whereas the living quarters or most frequent visited zones should have a normally open damper.

## 2.7 Sensors

The main and secondary control units will both contain sensors for variables that will be vital for the adequate function of the entire HVAC system. The main controller, inside of the house will contain a temperature and humidity sensor, that must be accurate within a degree for realistic temperatures, as well as a CO2 sensor that can be accurate within 100ppm. The temperature and humidity sensors will be used to determine, in accordance with user settings and controls, when to turn on and off which system in the HVAC system. The carbon dioxide levels will be used to adequately determine air quality. The secondary microcontroller needs to only have a temperature and humidity sensor, as the carbon dioxide levels outside the dwelling is data that is not sufficient to the operations of the HVAC system.

### 2.7.1 Temperature/Humidity

There are many options for temperature and humidity sensors. Getting the two together in a combined IC is possible and could save footprint and limit the parts. Most of these sensors output an analog signal which must be then converted to a digital signal in order to be processed and analyzed. These numbers are then usually formatted in order to be used in a reasonable way.

The sensors need to be easily powered by the device hardware already on the board and needs to have its data easily read and transmitted to the microcontroller for processing. This will be directly connected to the microcontroller via one of its ports for communications.

The secondary unit will likely have the same temperature and humidity sensor in order to prevent variables of discrepancy in retrospect of the two units. The secondary unit, though, will be in a much harsher environment outside than the main unit, so the possibility of choosing a different sensor exists. Further, the secondary unit will not, generally, have a direct connection to power, so power consumption on the remote unit becomes much more of an issue. The data from the unit will either be sent wirelessly, likely by and XBee chip, and also needs to implement the possibility of wired transmission just in case wireless transmission is not feasible due to thick walls or some other unforeseen barriers. Because the signals transmission will likely be digital and the remote unit will be operating on the smallest power possible, it is important that the signal is be given to the secondary remote controller digitally. All conversions will be done on the main microcontroller.

### 2.7.2 CO2 Sensor Applications

Sensors used in the previous version of the HVAC system were simply used to determine humidity and temperature. The use of these sensors is pretty obvious, which was to monitor temperature to determine when to turn on the AC units and the humidity was used to determine when to instead use a de-humidifier. Our design also now plans to include one more sensors, CO2 sensors. The CO2 sensors have a couple of possible different uses in our design. The first application for the CO2 sensor for our project would be for monitoring air quality and preventing flashover in the event of a fire. Also since our HVAC system is intended for typical households like houses and apartments, and not industry the CO2 sensor can also be used to monitor where a person is in the house.

One possible use is to make sure the CO2 levels in the overall house stay at a safe level. In this possible implementation the goal will to be monitoring air quality however it also ties into detecting fire and preventing flash over which will be discussed in further detail later. The goal is to have the sensor detect unsafe levels of CO2 in the overall house and when detected, either initiate venting of the house, or alert those in the house hold sort of alarm – be it via the on screen interface of the graphical user interface or via an audible alarm – or both.

The Center for Disease Control (CDC) website states that the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) states that indoor air quality should not exceed over 700 parts per million (PPM) over the outdoor CO2 levels. This is unrealistic for our design as we do not plan on implementing an additional outside unit as this could be cost prohibitive for budget as well as a cost manufacturability standpoint. So another air quality measurement the CDC is 1000PPM indoor CO2 concentration at which point the CDC considers ventilation inside to be inadequate and potentially unhealthy. Designing for that criterion we decided that a CO2 sensor up to 2000PPM is more than adequate for our design. This also happens to be the range of many so called “lower end” CO2 sensors that are available and help to keep our costs to a minimum. This sensor can also be helpful in detecting fires and helping to prevent a small fire from getting much larger. One potentially misunderstood and defining characteristic of building fires flashover. While many have been told in their life that they should “Stop, Drop and Roll” if they or their clothing is ignited, with the idea that starving the flames of air – really oxygen which feeds the fire – the fire will go out, the idea of preventing fresh air from coming into the building in the event of a fire is a misguided logical extension of this concept.

Flashover is defined by Merriam Webster’s dictionary as the “the sudden spread of flame over an area when it becomes heated to the flash point”. The flash point occurs when a given object or material is heated to the point at which it can suddenly ignite when exposed to fire. This is an unfortunate but common characteristic of many house and building fires. When firefighters actually vent buildings that are on fire to cool the building as outlined in the. What we plan on implementing is the venting operation that could be started as soon as the fire is detected by our system – defined as the CO2 sensor level being maxed out at 2000PPM for a certain period of time – and activating the venting of the building as well as bringing an alert up to the user in case this is an error and for diagnostic purposes. By venting the building as soon as a possible fire is detected the building can be cooled and prevented from reaching its flash point before firefighters even arrive on the scene.

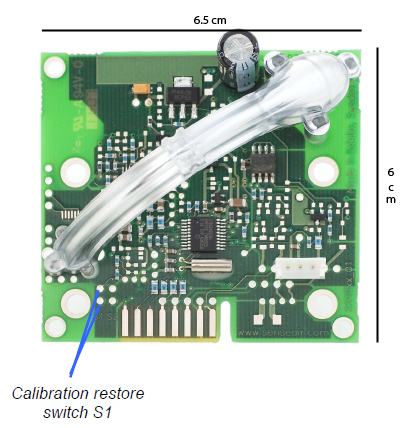
We plan on using the CO2 sensor to do this primarily for cost and because many forms of combustion produce CO2 which is a good detection of a fire. We caution however that this feature is just the basic functionality of this concept. Further research should be done by an expert to make sure the design of such a system is implemented in such a way that it is properly detecting a fire situation and at what point in time it should be activated in the detection of such a fire to properly cool the building. We plan to implement the basic functionality to achieve a fully safe and operational system.

The secondary possible use of the CO2 sensor is in remote units for detecting levels of CO2 in a particular “zone” or room of the building targeted for installation. When a person in the household enters a room or zone, CO2 levels will go up as a person exhales CO2. This is a good way of determining where – in what “zone” or room – the person is in the house. Ideally we see this being most useful if the sensor is placed in bedrooms. Combined with scheduling and/or just basic time of day information, the main control unit will be able to determine that it is bedtime (either scheduled or via default settings based on time of day) and a person or person(s) is in the bedroom(s (or a particular “zone”). Upon those two conditions being met the main controller would then tell the secondary microcontroller to redirect most if not all air flow only to the bedroom when it turns on. This would be done via dampers. The design should save a lot of money at night because instead of cooling the whole house, the AC unit would only be cooling the area of the household the person was in. Ideally we would be able to design these remote units to be powered by battery – similar to our existing remote temperature and humidity sensors – however this will be limited by the CO2 sensors themselves in their power requirements as well as how often we need to poll the sensors for readings on CO2 levels for them to be accurate and functional in implementing features. The more often we need to power on the device to get a reading for our implementation, the lower the battery life and the less effective the implementation will be.

For our project we will implement the CO2 sensor at the main controller. At this location we will use the CO2 sensor just for monitoring air quality for the dual purpose of health benefits and fire safety.

#### 2.7.2.1 Senseair® CO2 EngineTM K22-OC

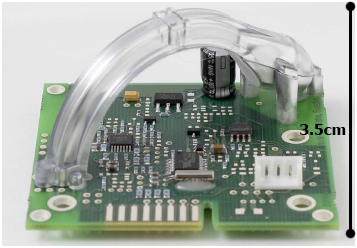
Senseair® offers both OEM and all-in-one pre-packaged designs for CO2 Sensors. One of the sensors we considered for implementing the CO2 Sensor requirement of our design was the Senseair® CO2 EngineTM K22-OC as seen below in Figure 17.



**Figure Figure 17 Senseair® CO2 EngineTM K22-OC Module**

(Reprinted with Permission from Senseair ®)

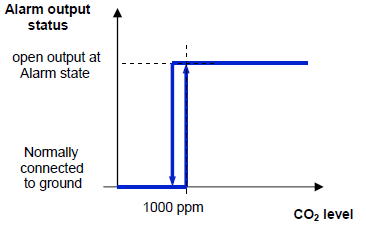
We considered their entry level CO2 sensor for our design as it is more than adequate for our design specifications for air quality. We do have a bit of an issue with the height of this unit should it be used on a remote unit. The data sheet lists the dimensions at a height of 3.5 cm which is larger than the other two modules we considered that were made by Senseair® by at least 40% (when compared to the K30 height of 1.4cm). This will definitely add to the remote unit height as all other components in the remote unit will be minimal compared to the height dimension of the Senseair® Module. This added device bulk, while not prohibitive to design for, may make the remote wall unit seem “unattractive” to the user and thus so long as we are within budge we will consider using other units despite being somewhat more expensive. The K22-OC is pictured below in Figure 18 and Figure 19 depicts the dimensions of the unit.

****

**Figure 18 Senseair® CO2 EngineTM K22-OC Module**

**(Reprinted with Permission from Senseair** **®)**

The Sensor is programmed to measure CO2 levels at an interval of 2 seconds, which is within our design criteria. CO2 high state is also triggered by a CO2 level of 1000PPM or higher – as seen below in figure 20 which also happens to be what is recommended by the CDC for being an unhealthy level of air quality.



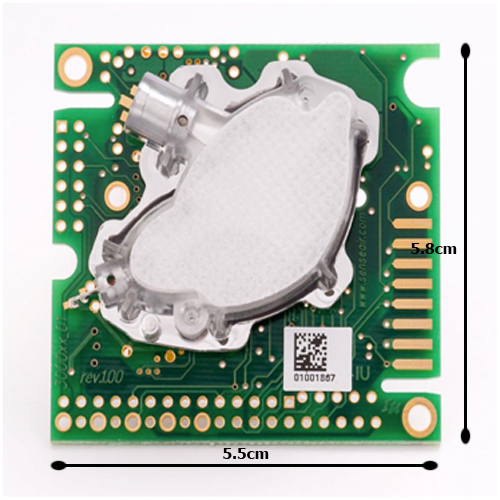
**Figure 19 Senseair® CO2 K22-OC: Depicting alarm status of CO2**

**(Reprinted with Permission from Senesair** **®)**

Once triggered the sensor will remain in alarm status for a preprogrammed 32 second interval giving amble time for the microcontroller to read the state especially if ever implemented on a remote unit where the frequency of polling the state of the CO2 sensor is important for battery savings. The device will also trigger on the alarm output – or open collector - in low power or if it detects a fault. Communication of this parameter can be done a various number of ways however for our design implementation that can be either I2C or UART. Also due to the way this chip is designed, our particular implementation this simply means that the communication of the state of the device is seen by our microcontroller to either alarm status on or off which corresponds to either a one or a zero value. This doesn’t give very much accuracy in terms of what the actual levels of CO2 are in the system, as its resolution is either above 1000PPM or below 1000PPM. Considering we were planning on using being able to read the particular numerical value of CO2 in PPM as a feature of fire detection device, choosing such a device would have to mean the fire venting feature would have to be cut or implemented in via another method.

#### 2.7.2.2 Senseair® CO2 EngineTM K30

The Senseair® CO2 EngineTM K30 is a much more advanced module than the K22. The measurement range on the K30 is 0 to 5000PPM with a 20 second response time and an accuracy of ±3%. The exact value measurement can be communicated via UART or I2C to a microcontroller. From a usability standpoint our design now can display an actual value to the user via the graphical user interface as oppose to the K22 which either has essentially to states, alarm on – over 1000PPM concentration, or alarmed off – under 1000PPM concentration. Also the potential for this chip to implement our fire venting feature is possible. This chip, as well as the K22, is designed to be built into stationary duct work. The K30 is pictured below in Figure 20.

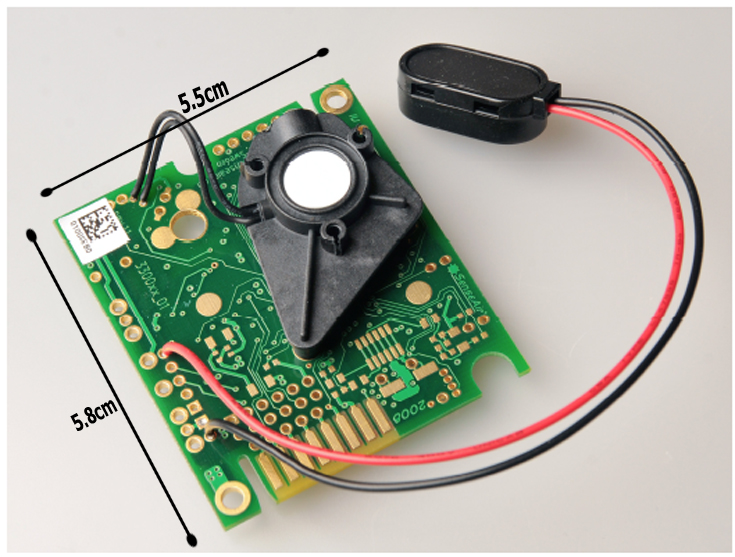


**Figure 20 Senseair® CO2 EngineTM K30 Module**

**(Reprinted with Permission from Senseair ®)**

#### 2.7.2.3 Senseair® CO2 EngineTM K-33 ELG/BLG

For remote sensing one particular unit seemed best suited for such an implementation the Senseair® CO2 EngineTM K-33 ELG/BLG. The Senseair® CO2 EngineTM K-33 ELG/BLG allows for us not to only monitor CO2 levels with minimal power use, but also monitor temperature and relative humidity all in one unit which should allow for a slightly more energy efficient design over a separately powered module for the temperate and humidity. The humidity and temperate is measured by a separate chip on the module Sensirion SHT11. This is also the same company that was used for the main board and remote unit temperate and humidity sensors of our existing designs. The power saving features of the sensor comes its ability to measure CO2, humidity and temperate and store its last measured value(s) with the time the measurement was made at – due to having a built in real time clock – in non-volatile memory. The memory is capable of storing a maximum 5400 logging points for all three sensor measurements are stored with timestamps. The K-33 is pictured below in figure 21.



**Figure 21 Sensair® K-33 BLG powered via battery**

**(Reprinted with Permission from Senseair ®)**

A microcontroller programmed to read the latest temperate settings will essentially be reading the values stored in memory. Communication with the microcontroller to read the values stored via in the memory can be done via the same communication methods as the previously mentioned Senseair® chips, via I2C and UART are both options. Battery power design is also flexible as the device has a relatively wide range of accepted voltages, 4.75 to 12 volts of DC. At one measurement an hour the device consumes approximately 250μA of current. The device has similar accuracy and range of CO2 sensing of the previous Senseair® K30. The range is 0-5000 PPM for CO2 levels. The relative humidity and temperate sensors on didn’t exist on the other modules we considered so they aren’t comparable. The range of the sensors however are within our specifications and measure temperate from 40˚C to 60˚C and within the accuracy specification of ±.5 ˚C. Relative humidity is the full range of non-condensing humidity, which is 0 to 100% with an accuracy of ±3%.

#### 2.7.2.4 Cost

For our application and budget CO2 sensors are somewhat cost prohibitive in our design standpoint of wanting to have both a main CO2 sensor and a remote CO2 sensor. The cheapest sensor we considered in detail for our design was the K22-OC. The chip is available from various 3rd party suppliers at the time of this writing at a price of approximately $99. However this excludes such parts as an I2C to USB cable for development purposes. The additional cost of that module at the time of this writing from 3rd party suppliers is approximately $39. This brings the total cost of prototype implementation at $140. The K30 sensor has the additional features we outlined above and can be found with the $39 cable for approximately $199. We feel this is the better deal.

For what should be essentially a remote thermostat, the Senseair® CO2 EngineTM K-33 ELG product is inhibitory for implementation due to cost. The K-33 ELG can be found online for $299 plus the additional cost for the $39 I2C to USB cable for rapid prototyping, bringing the total cost of this development environment to $340. From a production standpoint, buying these devices in bulk should result in a lower overall cost from the developmental costs. This however is not a focus of our research.

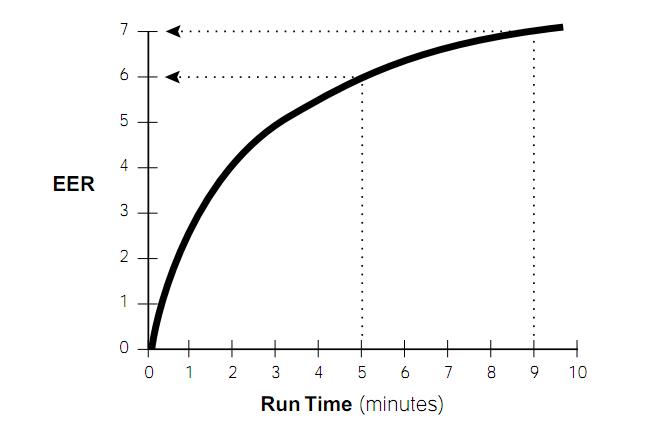
## 2.8 Seer Ratings

A large portion of the goal of this project is to provide a more energy efficient means of cooling and heating a house. Aside from hooking up a test unit and factoring energy spent through trial and error, a method was researched to be found to quantify energy savings using a smart HVAC system. SEER stands for Seasonal Energy Efficiency Rating for central air conditioner unit. These ratings are determined under laboratory test to determine performance of evaporator coils and condensers. The seer rating is given as percentage based on outdoor temperature, as most air conditioners will most understandably work more efficiently at a lower outside temperature, there is percentage differences in their performance. The most common misconception with air conditioners is that bigger is better. This was the purpose in the seer ratings. Larger air conditioner units take longer to reach their maximum efficiency. Energy consumption is at its largest in an air conditioner as it is approaching its maximum efficiency, not at its peak. A larger unit may be able to cool a house in shorter time, but it never reached its maximum efficiency. A smaller unit will be able to more economically cool a house often, depending on outdoor temperatures and indoor temperature settings. In circumstances where a three ton unit were to cool a house in a given time but never reach peak efficiency, a two ton unit could have cooled the house at a less detrimental effect to the power usage of a house.

House air conditioner systems are commonly given sizes in tons of cooling. One ton is equal 12,000 BTU’s, or 3,500W. One ton is the approximate power required to melt one tone of ice in 24 hours. The annual cost of running of an air conditioner unit is given by:

This will give the overall money spent in a year on an air conditioner system. This equation will be transferred to a monthly value and displayed on the user interface. The difference in seer value will be subtracted from when a smaller unit is used in place of the larger rated unit. If the cooling time was less than the larger units maximum efficiency time, then the amount of time differenced will equal the theoretical energy savings. Part of the installation of the system will require the knowledge of the seer rating of the two units actually installed, and the seer unit of what would have been installed in the house (generally the size of the two units added together). There will be other cost saving additions to our power saving section in our code that will include energy consumption caused by not having an intelligent thermostat timing schedule.

The process of the air conditioner turning off before its maximum efficiency is reached is called short cycling. The overall efficiency of an air conditioner is low when it is initially started. Maximum efficiency is approached in approximately ten minutes, depending on the unit. The EPA Energy Star program rates energy efficiency rating.Figure 22 below shows the graph developed.



**Figure 22 Short Cycle Energy Loss**

Every air conditioner gets rated by the EPA for energy star compliance and given a Seer rating. A Seer ranking of 13 is minimum requirement by the United States now. A rating of 14 is twice as efficient as a rating of 7. Further beyond efficiency, it has been found that the life cycle of the air conditioning unit is negatively affected by short cycling. (Mulroy, 1986) A combination of the Seer ranking and the short cycle times are important. The system will have the opportunity for the installer to set up the system to input the seer rankings of each unit as well as the minimum run time to maximum efficiency. The air conditioner will not turn off until its maximum efficiency is reached. This will cause the house to occasionally get lower than the desired temperature, but this will just mean the house will take longer to reach a point of needing to turn back on.

## 2.9 Serial Connections

#### 2.9.1 UART

An universal asynchronous receiver/transmitter (UART) is a serial Input/Output (I/O) device that allows for data to be transferred from a parallel format to serial format. A device with a UART chip embedded can transmit data as a transmitter to another UART equipped device who receives the data as a receiver but the roles can also be switched at any time once one starts the process by transmitting data. Bytes are transferred bit by bit in a sequential manner and repackaged by the receiver into bytes. During this process, the shift register which is key in shifting the bits together to create the associated bytes initially transmitted by the transmitter. Data in the form of characters are transmitted as bits to be sent to the receiver. The entire transfer frame consist of one start bit, eight data bits, and one stop bit. The start bit is always active low which puts its value at '0' when data is being sent and received by the receiver. The stop bit is always active high which put its vale at '1' when data is being transmitted. The Figure 23 below shows the UART Transmission Character frame.

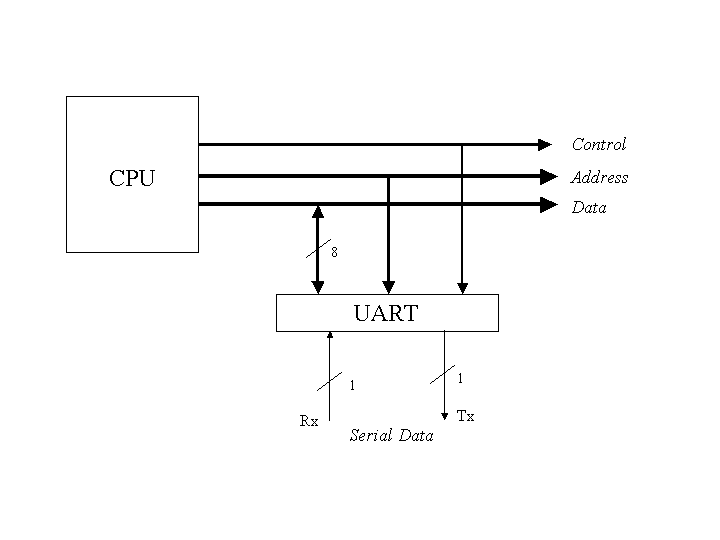
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Start | Data  0 | Data  1 | Data  2 | Data  3 | Data  4 | Data  5 | Data  6 | Data  7 | Data  8 | Stop |

**Figure 23 UART Transmission Character Frame**

UARTs were first designed by Gordon Bell for use with Programmed Data Processors (PDPs) which were minicomputers made by Digital Equipment Corporation in the 1960's. UARTs typically contains a clock generator, input and output shift register, transmit/receive control, and read/write control logic. Optional features in a UART are transmit/receive buffers, parallel data bus buffer, First In, and First-Out (FIFO) buffer memory. The UART's clock generator allows for multiple bit rates to be handled so that data could be sampled at any increment of time for instance in the middle of a bit period. Each bit lasts as long as the set clock pulse.

UART have become well known over the years due to its association with communication standards such as RS-232 RS-422 and RS-485. Recommended Standards (RS) 232 was first introduced in 1962 by the Electronic Industries Association (EIA) to be used with digital data communication among mainframe computers to remote terminal during this time period. Over the years, serial communication has progressed beyond just general purpose large scale computers to microcontrollers and even portable devices. For years, RS-232 ports have been used as a standard part for serial communication in devices such as modems, printers and computers.

There are four errors that can occur with using UARTs. An overrun error is when an incoming character to the receiver is lost due to the fact that the CPU did not keep track of the input buffer size for the UARTs and it becomes full. An Underrrun error occurs when data is sent by the transmitter but the transmit buffer is empty which implies that the data has been completely sent and nothing remains. This error occurs more commonly in synchronous systems. Framing errors occurs when the data line is not in the idle state and the start bit and stop bit are not invalid to identify the beginning of a new character transmission. Parity error occur with Universal Synchronous/Asynchronous Receiver/Transmitter (USART) when the parity bit is enable. When this is active, and the received data is not the expected value being received a parity error occurs. Figure 24 visualizes this buffer.



**Figure 24 UART Transmission with CPU**

#### 2.9.2 I2C

An Inter-Integrated Circuit (I2C) is a multiple master- serial single-ended computer bus invented by Philips in the early 80's to provide communication between low speed components on the same circuit board. I2C consist of only two bidirectional open-drain lines, a Serial Data Line (SDA) and Serial Clock (SCL) which are connected to the voltage source by pull-up resistors to allow the inputs of the logic system to settle. There are 16 reserved addresses and each has an address space of 7 bits so the maximum 112 bits can be communicated on the same bus. Common speeds of the I2C bus are 100kbit/s in standard mode and 10kbits/s low-speed mode. The bus is also made up two types of nodes which are master and slave. Master node is the node that controls the clock and addresses slave nodes. The slave nodes receive the clock input and addresses from the master. Since this bus is a multiple master serial single end bus, this means that there can be more than one master.

A bus can either be a master or slave and consist of two modes of operation for each role. Master transmit is when the master node is sending data to the slave, but master receive is when the master is receiving data from a slave. Slave transmit is when the slave node is sending data to the master, but slave receive is when the slave is receiving data from a master. In the initial phase of communication, the master is in master transmit mode where it sends a start bit and the bit address to signify which slave is desired followed by a single bit 1 if writing to or 0 if reading from.

If the slave exists, an Acknowledge (ACK) bit for that address would be sent. The ACK bit is also active low (0). The master then proceeds in either transmit or receive mode based on the previous bit sent after the address, and the slave operates in the mode required. Both address and data bytes are sent with the most significant bit first. The start bit is indicated by a high-to-low transition of SDA with SCL high; the stop bit is indicated by a low-to-high transition of SDA with SCL high. If the master is in master transmit and the slave is in slave receive, bytes of data is transmitted with the slave responding with an ACK bit. If the master is in master receive and the slave is in slave transmit, bytes are received from the slave with the master sending an ACK bit after every byte except the last byte. The master either sends a stop bit to end transmission, or another START bit to retain control of the bus for another transmission.

There are three basic types of messages all of which begins with a START bit and ends with a STOP bit. The first is a single message where a master writes data to a slave. The second is another type of single message where a master reads data from a slave. The last one is combined messages where a master issues a minimum of two reads and/or writes to one or more slaves. The figure below illustrates the process of data transmission within the I2C bus. The 'S' signifies START bit and 'P' signifies STOP bit. The blue sections indicate that SDA sets the transferred bit while SCL is low and under the green the data is received when SCL goes high. When the transfer is complete and a STOP bit (P) is sent, the data line is pulled up while SCL remains constantly high. Figure 25 shows how data is transmitted through I2C



**Figure 25 I2C Data Transmission Timing Diagram**

#### 2.9.3 SPI

The Serial Peripheral Interface Bus (SPI) bus is a duplex, synchronous I/O device named by Motorola that allows for serial data transmission between the CPU and other devices. In this data bus, devices communicate in two modes master or slave in which the master initiates the data frame. Multiple slave devices can be communicated with through the use of slave select lines. Sometimes SPI is commonly called a four-wire serial bus. The four logic signals within a SPI bus is the Serial Clock (SCLK), Master Output, Slave Input (MOSI or SIMO), Master Input, Slave Output (MISO or SOMI), Slave Select (SS).

In the initial phase of data communication, the master first configures the clock, using a frequency less than or equal to the maximum frequency of the slave device's clock. Common frequencies are in the range of 1–70 MHz The master would then set the slave select low for the desired slave.

During each clock cycle, a full duplex data transmission occurs in which the master sends a bit on the MOSI line then the slave receives it from that same line or the slave sends a bit on the MISO line then the master receives it from that same line. Data transmissions generally involve two shift registers of a given word size of eight bits, one in the master and slave; and are connected. The most significant bit is shifted out first, while shifting a new least significant bit into the same register. After that register has been shifted out, the result would be that the master and slave's register values were exchanged. The next step would require the devices reading or writing the values to memory. The process is repeated if there is more data present. Transmissions may take up any number of clock cycles. When all of the data is transmitted, the master stops toggling its clock. Finally, it deselects the selected slave(s). The Figure 26 below illustrates the transmission of data from slave to master.



**Figure 26 Data Transmission from Slave to Master**

# 3 Design

The following section discusses the actual design of the smart HVAC 2.0. All the information presented in the following section will address the actual ideas used. These design specs were changed slightly from the original paper, of which sections were noted in research section and all design portions made it to final production.

## 3.1 Software Design

The high level controller software design is based on flexibility and simplicity. In order to meet our specifications, the system is able to communicate back and forth to the main control unit which will also interact with the wireless unit. The high level controller provides users with the medium to adjust the system constrains in regards to its inputs and outputs.

At this layer of the system, the system operators are separated into two groups which are customers and technician. Customers have the ability to adjusts system features to obtain the appropriate level of comfort. Technicians have the same abilities as customers but are also allowed to enable system components through the use of an administrator mode viewer. The levels of control are illustrated below in Figure 27.



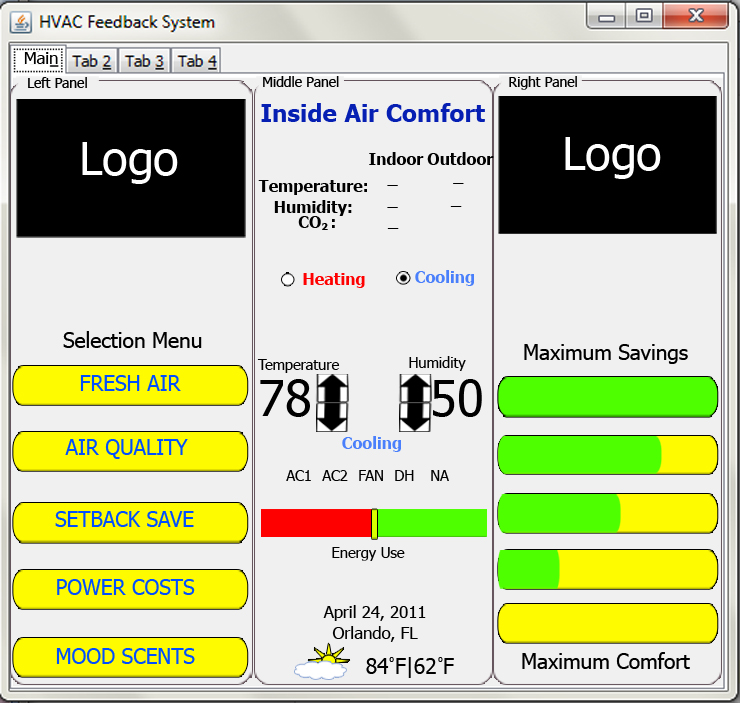
**Figure 27 Use Case diagram for HLCU**

The software for the high level system was written in the Java programming and consists of five classes at this time. The choice of five classes decided upon with the purpose in mind that dividing the system into modular divisions the program could then be coded separately and allow for efficient debugging if needed. The HVACappFrame class creates the window display and the tabs for the application. The Main\_View class constructs the panels, buttons and methods for the Main tab on which the user will do most of their interaction. In the Scheduler class, a table will be constructed to store weekly customizable system settings' entries which will allow for reduced interaction between the user and system. The User\_Settings class allows for users to edit the location for the weather status RSS feed and the scheduler status. Final class is the AdminClient which consist of a login window for technicians or qualified personal to use to enable or disable components in the system. Figure 28 below illustrates a class diagram of the high level control system.



**Figure 28 Use Case diagram for HLCU**

The "Main" tab on the user interface for the High level control unit User interface is the most interactive part of the system due to its multi-feature. In the Left Panel, users are allowed to select and customize system features with the five provided Selection Menu buttons. In the Middle Panel, the indoor and outdoor temperature, humidity and carbon dioxide (CO2) are displayed. Users are also allowed to select whether they would like to heating or cooling related settings. The enabled components are listed below the temperature and humidity selection area. The energy consumption gauge which represents the energy utilized by the system is also displayed. A new proposed feature to the system is the weather notification area at the bottom of the middle panel. This new feature helps to give the user an increased awareness of the weather to help with temperature scheduling and system usage. The Right Panel consists solely of the Power and Comfort level buttons that gives users a range of five comfort settings. The new design for the main tab is illustrated below in Figure 29.



**Figure 29 Proposed design for the   
HLCU User interface Main Tab.**

The main controller's software design focus is based on simplicity and customization. In order to meet our requirements, the system readily displays the indoor and outdoor temperature and humidity, sends the appropriate values to the main controller unit to control multiple relays to power HVAC components and controls venting modules. In addition to the automated controls of the main control unit, it has a graphical user interface to allow a prospective user to adjust system control settings.

The interaction involved in the system is depicted as a use case diagram that handles all of the HVAC controls. The operational conditions of the system will be passed and validated by the main control unit based on the adjusted user settings. These settings changes adjust the power relays and the venting systems when changed. The control process of the entire system also receives reading from a wireless temperature and relative humidity sensor. With the provided reads the settings that the user adjusts can be used to achieve their desired comfort and power levels. Figure 30shown below is the use case diagram for the unit's control.



**Figure 30 Use Case diagram for MCU**

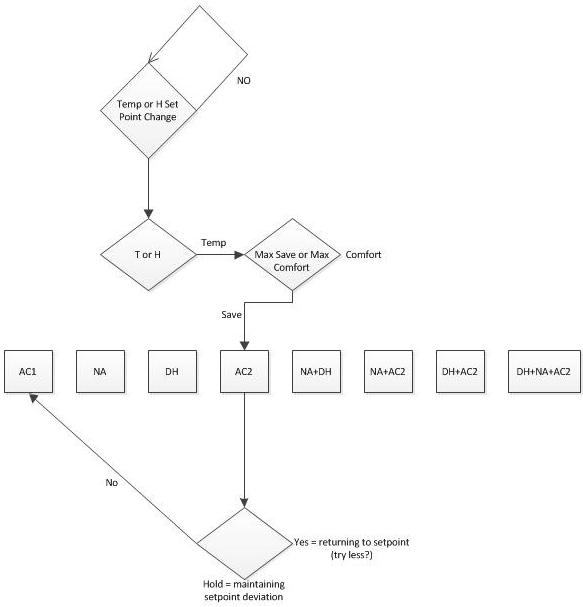
The software design for the wireless sensor is more basic than the design of the main control unit. The wireless sensor will transmit a response for request by the Main control unit for information on the temperature and relative humidity. When this operation is executed, the system will gather the sensor reading for transmission. This information is transferred back to the MCU. Figure 31 illustrates the use case diagram for the wireless sensor.



**Figure 31 Use Case Diagram for the wireless sensor**

### 3.1.2 Embedded Software

The embedded software portion of our project is the core of this design as it determines the functionality of the system. Having this working alone enables us to receive full credit for our senior design project. Everything in our project really relies on the system being able to perform the tasks of interfacing to all the components. Our sponsors have specifically indicated that the main feature to this system should be the power efficiency. Our embedded software design has really started to increase in complexity due to the definition of power efficiency given by the sponsor. Compared to version 1, the embedded software in version 2 is going to be a lot more sophisticated and robust since there are a lot of more situations that need to be covered. Further, the math to determine when the air conditioner units go on was adjusted. Version 1 just adjusted temperature when the ranges got irritated. If the house was hotter than the air conditioner setting, then the unit would turn on until the temperature was adjusted. Instead, short cycle considerations will be added to the logic of the version 2. If the user decides, the air conditioner will have a minimum operation time of a short cycle. Though the unit may run longer, its frequency of turn on should be greatly reduced. The amount of savings accrued by doing this is mathematically calculated and displayed in a per month basis. The user will only have to adjust the dollar amount of average price their bill has per kilowatt hour, and the system will calculate the estimated cost savings of this unit vs. either the typical unit installed for that house or the old system it was replacing. The cost is factored by both the difference in price of operating a smaller unit (when applicable) vs. the larger rated unit, further the energy cost of limiting the short cycle energy inefficiency. Seer rankings vary per unit depending on outdoor temperatures, so depending on the outdoor temperature readings that the secondary control unit has will also further decide the proper air conditioner unit to turn on, instead of just relying on the user power saving adjustments. Figure 32 below is a graphical description of the basic functionality of our embedded software:



**Figure 32 Basic Functionality**

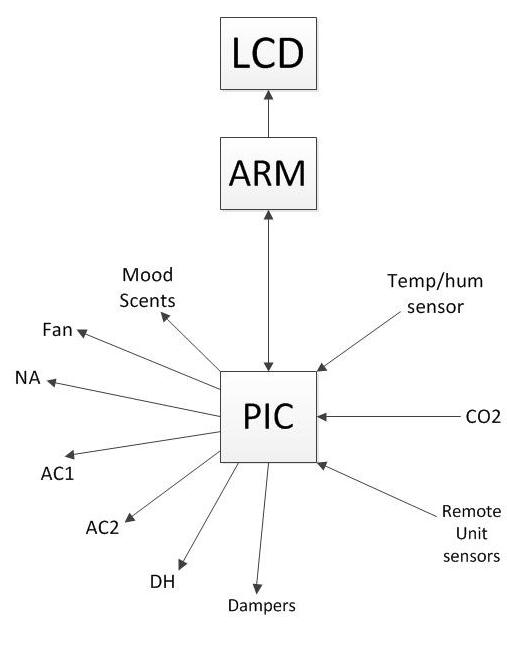
#### 3.1.2.1 Components and Functionality

According to the figure above, we see that the functionality of the microcontroller is complex but we believe as a group that it is feasible with our programming abilities. First the user gets to do a temperature or humidity set-point change. If there isn’t a request for change, then the systems stays in that state until there is a “YES” for a set-point change. When the user has requested a set-point change, we go to the next state where the machine decides whether it’s a temperature change or a humidity change. If there is a humidity change, then the system will activate the humidifier by flipping the switch on the relay. The way this is done is that the microcontroller will take away the 3.3 V that it sends to the relay by default. If there is a temperature change, the system will turn on AC2 which is the smallest ton unit in any combination of components that the user have in their system. Based on the power settings of the user, the microcontroller will wait for a certain amount of time to determine the actual temperature and desired temperature to decide whether the AC2 alone can do the job of cooling or more components are needed. For example, the user wants Maximum Comfort at 70 degrees Fahrenheit. The house is at 80 degrees Fahrenheit and outside is 95 degrees Fahrenheit. In this example, the microcontroller will realize that outside is hotter than inside, so natural air system won’t be turned on and used to cool the house. It will instead turn on AC2. Since it is in the state of Maximum Comfort, it will wait approximately 5 minutes to check to see if the house has gotten down to 70 degrees. If the house is not at 70 degrees, then it will turn on AC1 which is the bigger ton unit. If AC1 cannot do the cooling needed by the house, then it will turn on both of them to be able to cool the house. All this is done with delays to make sure that the user is getting the most out of the system while being power efficient. On the other hand, if the user wants maximum savings instead of waiting for 5 minutes to check to see if AC2 has cooled the house, it will wait 20 minutes to see if the house has cooled to the desired temperature. If it hasn’t then it will follow the exact routine as explained earlier except with a delay of approximately 20 minutes. In essence, the system is going to be designed to exhibit adaptive characteristics. All of this is also dependant on the selected active air conditioners seer ranking and their maximum efficiency minimum time cycle. The minimum time of each unit to be turned on may be chosen to meet minimum operating times in order to reach its maximum efficiency. This will be part of the user input and will not be mandatory, but should be set to default as it does save energy cost as well as extend the lifetime of the unit.

In the figure provided the components of the system are clearly stated. The system has the components of Natural Air System (NA), AC2 (the smallest ton unit in the system), AC1 (the biggest ton unit in the system) and Dehumidifier. Additional components that exist but are not included in this figure are the Fan, mood scents and the dampers. As it is seen based on the example, we may have multiple components on at a time. Some of the suggestions from the sponsor were to have the mood scents activate anytime the fan comes on for 15 seconds. The main difference in this version 2.0 compared to version 1.0 that the sponsors have keyed in on is adaptive nature of the system. Compared to version 1.0, version 2.0 is able to run in a home that doesn’t necessarily have all the components installed. In version 2.0, if the consumer doesn’t have enough money to buy all the components recommended by the company, they can still have the system with just the components that they have. In this case, this system has gone from not just a power efficient system, but also a budget efficient system. This is implemented by allowing the technician, through the application GUI, to indicate to the system what components are installed in the system and those that are not in existence. Our code accounts for situations when a component doesn’t exist and takes this into account. This is where the adaptive feature is in place in our system. The sponsor has made it clear to us their intentions of attracting their customers by giving them the flexibility to buy additional components to be able to add additional cost savings to the customer. The Natural Air system is designed to allow the house to cool without the AC units if the temperature outside is cooler than the temperature inside the house. For example, if the user wants their power settings at maximum savings with a temperature of 70 degrees inside and the temperature outside is 60 degrees, then the system will simply chose to use the natural air system. This is the most effective method of saving money since there would be no need to run any of the electrical equipment besides the fan and the vent for natural air to be open. The dampers that we are including in the system will be used for zone control. This is another method of saving the user money that has to be determined by the system. Therefore, when the system sees that it is evening, and it has to cool the house, it can know that the only room to cool should be the bedroom and therefore will close the vents everywhere else using the dampers. Mood scents are used to aromatize the desired area. This is done when the fan comes on, which can be turned off by the user and manually set for once every set amount of time. Overall the embedded software is well involved and requires the input of all four of us since there are so many states and conditions to account for.

#### 3.1.2.2 Microchip and ARM

We have chosen to have the PIC as the microcontroller to interact in the environment and the ARM to be totally dedicated to the Graphical User Interface and webserver. Both of the processors are used in the embedded process but with different loads. The PIC microcontroller is the main one handling the load of the embedded software since it is responsible for reading the data from the sensors and activating the relays that are assigned to the components. The ARM on the other is used lightly in our embedded software since it is mostly there for our graphical user interface. The ARM passes values to the PIC to make decisions such as power settings and it will also be receiving signals from the PIC such as the temperature and humidity readings. We have decided to use UART/ RS-232 to communicate between the PIC and the ARM. Although we had concern with the two processors having time conflicts, we have been successful with using RS-232 communication port which is asynchronous. The RS-232 definitely works to our advantage as we don’t have to worry about synching the clocks of our processors. Below in Figure 33 is the hierarchy diagram of our system:



**Figure 33 Hierarchy**

#### 3.1.2.3 ARM Processors

The ARM processor, which is represented in the figure above, is between the LCD and the PIC microcontroller on the main control unit. The ARM processor is being heavily used for user interaction and sending signals to the PIC plays an important role in the functionality of our embedded system. The ARM is on both ends of the communication which is transmitting and receiving. As we stated earlier, we are using the RS-232 communication for these two processors to communicate with each other. The ARM also drives the LCD via HDMI allowing us to have the RS-232 port for the PIC. The arrows in the diagram indicate the direction of communication with all these devices. The ARM transmits and receives data from the PIC microcontroller. The ARM receives data from the PIC such as the temperature and humidity data from inside and outside of the house. The ARM needs that data to implement in the GUI so that the user can know the temperature and humidity readings inside and outside the house. The ARM is also going to need to know the CO2 level in the house which once again will be provided by the PIC. The ARM transmits important information to the PIC that will enable the PIC to make decisions. The ARM has the initial information as to which components are in the system since that is determined in the GUI. In order for the PIC to work to its desired functionality, it needs that information. Therefore, the ARM is going to have to transmit that data. The ARM will also have the information as to what temperature and humidity set point that the user would like. In order for the task of turning on the components to take place, the PIC will need to know that information which must be provided by the ARM. The ARM also has the power setting level, which is critical to the functionality of power efficiency. This information is then sent to the PIC. Evidently, we can see that the ARM does play an important role in the functionality of our embedded system. The ARM does connect to the LCD, but all the interactions are with the graphical user interface.

#### 3.1.2.4 Microchip PIC microcontroller

This processor is really taking most of the sensor load of the embedded software in our system since it is the main microcontroller interacting with all the different components in our system. As is evident from the diagram, the PIC interacts with just about every component in the system directly and indirectly. The PIC communicates with the secondary microcontroller via XBee to get the information of the outside temperature and outside humidity data. That information will be then sent to the ARM so that the user can have a hold of that data. The PIC is also processing the data from the sensors on the main control unit which are going to provide the data of temperature, humidity and carbon dioxide. These data are sent directly to the ARM which then displays it via the LCD to the user in the java program’s graphical user interface. This is most of the inputting that is done to the PIC by the components of the system.

The PIC also does a great deal of outputting data in the system’s functionality. Once the PIC gets the information from the ARM it uses that data to make critical decisions that will enable the system to perform efficiently. Using the information from the ARM, the PIC is going to send a 0 or 1 (digitally) or 0V or 3.3V (analog) which then is going to go to an NPN, and then out to the relays. According to the spec of the relays, when it sees 0V from the processor, it is turned on, whereas, when it sees 3.3 V, it is off. Intermediate to the PIC and the relays are NPN transistors. These transistors are used to protect the microcontroller and also to buffer the output signal. When the PIC sends a 0 digitally, or 0V analog, the NPN is going to be in cutoff, which then puts a zero volts at the relay. Having this condition the relay turns on. Otherwise, if the microcontroller sends out a 1 digitally, or 3.3 V analog, the NPN will be on and therefore will give a 3.3V at the collector tied to the relay. In this case the relay will be off. Based on the number of components that could be possibly be implemented in the system, we designed with 14 relays to have total control of the operations we designed for and additional options that can be assigned later. While we assumed we could use the same PIC microcontroller that the previous group used, it turned out for our design criteria we needed a different PIC. We instead used the PIC24FJ128GA010. Regardless of what goes on under the hood in our systems, the results are shown through the functionality of the relays shutting off and coming on when they are supposed to. This is the heaviest part of our embedded software as it will be instructing the PIC as to which relays to turn on and off based on the analysis of the current situation. We designed the system that way since we want to devote the ARM’s resources to the graphical user interface and devote the PIC’s resources to the functionality of the system and components itself. It was uniformly felt by the group that if we couldn’t give the ARM the full control of the system due to the lack of I/O ports and we felt combining everything into the system even if it did have enough I/O ports might affect the responsiveness and reliability of the system. Due to the fact that we want to make sure that our system improves the overall performance of version 1.0, we have taken the time out to ensure that we allotting to ourselves enough resources so that the system can reliably work. We may have over engineered the computational power of the GUI but we feel like in order to consider this a success, we need to have a smooth and operable machine such that the user is comfortable using the device. We could then scale the system down based on its performance or should it be slightly unresponsive we could even scale it up. One of the biggest issues with version 1.0 was that it had a very annoying lag which we think could have been due to memory and programming issues fundamental to the design of the system. Our goal from the beginning was to make sure that our systems run smooth and that the graphical user interface is pleasant to the eyes and easy to use as this will attract customers.

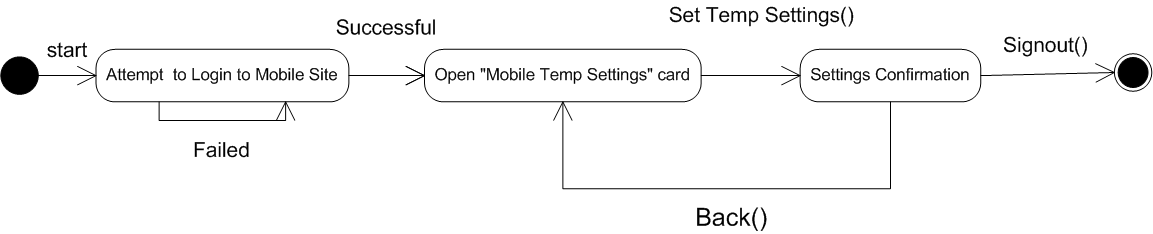
To conclude this section, it’s clear that this embedded software for this system is key to the system’s and the group’s success. Fortunately design around this, we have everything working, but if the embedded portion was ineffective, everything else fails and is useless. What guided our development was ensuring that this would work, and once we had this portion of the project done, we believe we can add all the “colors” to the system by adding the additional features such as WIFI connectivity and additional sophistications. Since we were able to get most of the core functionality of the system working well we were only able to add some of these additional features, and not as many as we had hoped.

3.1.3 Mobile Web Site

For additional access, flexibility, and control, a mobile website will be designed for customers or technicians to utilize for unit testing and initialization. The programming for the mobile web site was written in PHP and HTML with CSS for design. The web site will consist of a maximum of two lightweight pages for efficient usage with mobile phone or devices. This design decision was made based on the concept of giving users the most essential features without creating a busy page that may take longer to load if the device viewing the page isn’t powerful and also due to the limitations of mobile devices' screen sizes that typically range from 3 - 4 inches diagonally, we designed to fit to the limitation of these devices. Along with the essential features needed to have a comfortable experience, security is also a factor in designing this mobile page.

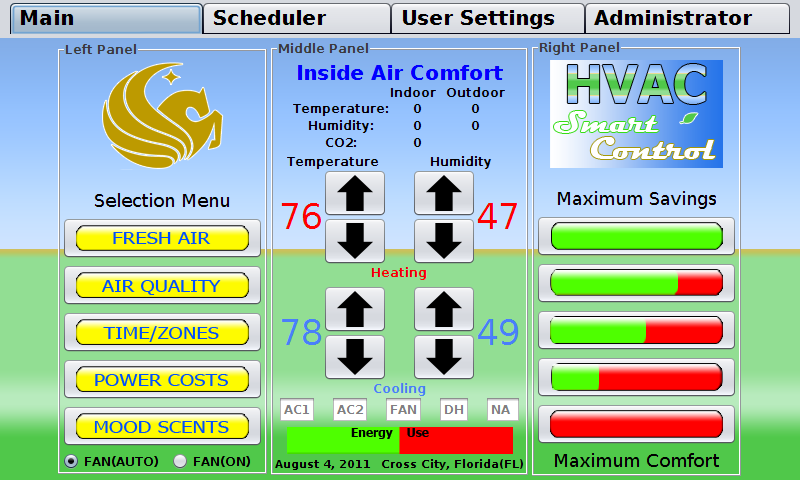
Ideally first page of the site would’ve consisted of a login page design that will have fields for username and password input and a submit button that execute a method that will validate the user's existence through communication with the high level controller. However this was unimplemented due to time constraints. However since the options to actually change values of the system, while code complete on the website programming, aren’t implemented in the Java program running on the ARM processor that will allow it to change settings. “Mobile Temperature Values” is instead of being the second page, the main page reached upon typing in the IP address of the computer. It is a simplified version of the Main Panel's "Middle Panel" shown in figure 28. This page displays the indoor and outdoor temperature, humidity and indoor carbon dioxide (CO2) level. The page can, upon a simple implementation in the Java program, have a functioning button to increment and decrement the temperature and humidity, the ability to change the option to select between Heating or Cooling and change the comfort levels/savings. All of the data on the site is accurate and current to approximately the latest 30 seconds of system sensor information make users aware of the latest reading of information with a time stamp that shows the last read value.

In formulation of this mobile web site, a state diagram is created to visualize the steps in which the user's interactions could be handled and the results of the selections made. A state diagram of the proposed implementation of the mobile web site is below in Figure 34.



**Figure 34 State Diagram for Mobile Web Site**

As stated before the mobile site's main page, "Mobile Temp Settings" page is a simplified version of the Middle panel on the High Level Control Unit. The layout of the window follows the same design layout, but has an absent of the appliances display, system energy use bar, date, and weather report for that day. Figure 35 below is a mockup of this page's design and its proposed controls.



**Figure 35 Main Layout**

## 

## 3.2 Wireless

There are two sets of wireless protocols. The main control unit and secondary control unit will talk to each other using the 802.15 protocol, via the XBee wireless units. The ARM display unit has a built in Wi-Fi chip for internet connectivity.

### 3.2.1 RF Communication with Secondary Unit

Though wired connectivity has been installed on the secondary remote microcontroller, the preferred method of transmitting data is through wireless. Because the unit is remotely installed, powering the unit was an issue as all devices connected to the secondary unit must be low power and not drain the power too fast. The need of low power lead us to using an XBee chip instead of going with the already installed 802.11 Wi-Fi standard that is installed on the ARM main control unit. XBee is a low power, easy to use interface for wireless data transmission. The XBee operates at 1mW of power. A supply voltage of between 2.8V to 3.4V is required to power on the device. This 1mW power drain leads to a real low current draw of 45mA to 50mA depending on transfer or receive mode. Its data bandwidth is and was more than what we needed for the amount of data being sent.

The XBee chips on both the main microcontroller unit and secondary remote microcontroller unit are both directly connected to the individual microcontrollers. Using UART communication to transmit the data back and forth between the microcontrollers and the XBee chips provides a buffer to help prevent data getting lost. There is not a lot of data transmitted between the two, and the data is not streaming.

### 3.2.1 WIFI internet connectivity

The ARM display unit Pandaboard has a built in Wi-Fi module. This board in particular is using WiLink 6.0 solution TiWi-R2 Module. This module is a low cost 802.11 b/g/n and Bluetooth embedded controller. This allows the unit a wide array of options to connect to the internet at the given location. The drivers for this device are also natively supported in the Linux kernel as of Linux kernel 2.6.38.8. Linux is notorious for its lack of driver support for certain devices and having it supported at such a low level provided us with more time on implementing more of our design criteria effectively.

## 3.3 Sensors

The secondary and main controllers both had to be design with the installation and interaction with a temperature and humidity sensor. The main control unit also has Carbon Dioxide sensor which can be implemented in future designs for the dual role of detecting whether or not someone is in a room – and thus determine what zone should be active in the house – however in our design implementations its purpose is to determine air quality and whether or not to vent fresh air in or ventilate the room to provide healthy air quality levels.

### 3.3.1 Temperature/Humidity

The temperature and humidity are measureable within realistic extreme weather temperatures. These have been discussed with the sponsor as 0 to 110F in Florida within 1 degree of accuracy. Humidity is read from 0% to 100% within 1% accuracy. Our design used the Sensirion’s SHT21 that accurately measures temperature from -40 to 125 C within 0.01 degree accuracy as well as humidity from 0% to 100% within .03% accuracy. Clearly this sensor was well within the scope of our projects specifications. It does the analog to digital conversion within its own IC and outputs a digital signal that requires a small amount of conversion to be applicable. The IC also allows us to avoid surface mounting the sensor. This sensor has a footprint of 3.3mm and is already calibrated. Sensors in our design that didn’t need periodic calibration and came pre-calibrated was a necessity to make sure our product was always reliable.

The sensor communicates using I2C protocol. This uses two bidirectional lines, one for clock, and one for data. This was more than adequate for the small data sent. The protocol initializes as a start command is sent from the controller to the sensor. This wakes up the sensor to receive a command. There are six general commands: temperature measurement, humidity measurement, temperature measurement with no hold master, write user register, read user register, and soft reset. In hold master the clock line is blocked by the sensor, whereas the clock line is left open in no hold master. The output from this is a 14bit digit with the two LSB’s describing the type of measurement. These digits are then sent to the microcontroller where math is necessary to convert the digits to a usable form. The following equations are used:

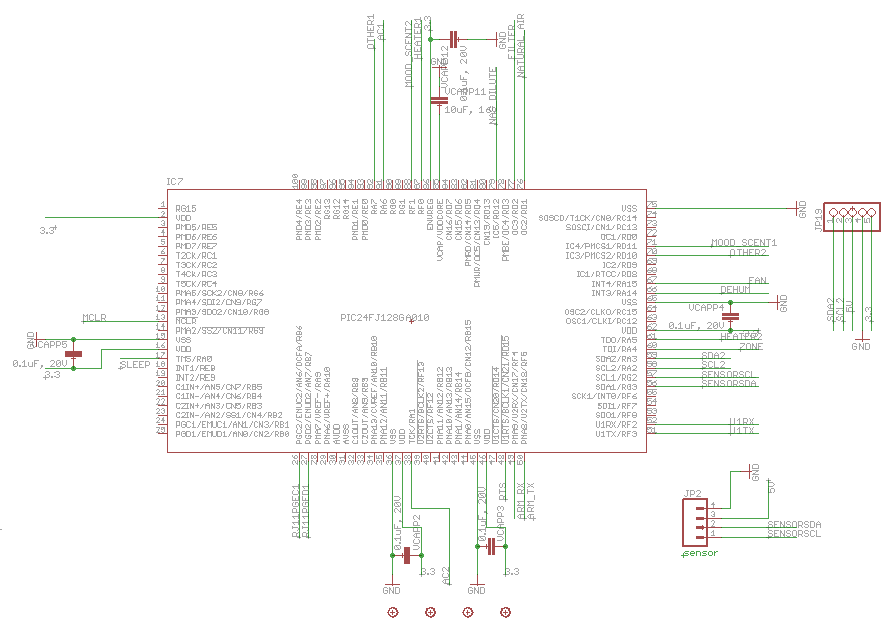
Where RH is humidity level, Srh is the variable transported from the sensor, T is temperature, and St is the variable transported from the sensor. The secondary control unit requires no CO2 sensor, so using this cheaper temperature and humidity sensor make sense. Many CO2 sensors we had researched had temperature and humidity sensors included on them, however we decided against this. The cost effectiveness of our design, while it is a prototype, is important as well and providing our own temperature and humidity sensor was cheaper.

3.3.2 CO2 Sensor

While we were considering using a CO2 sensor on an additional inside remote unit, for cost reasons we decided against this. While we did design for both, we felt the feasibility of using the additional inside remote sensor with CO2 sensor for purposes of determining whether or not a certain zone was active based on the CO2 levels – with a person in the room raising the CO2 levels – was not a well thought out design. A motion sensor would probably be more practicable and cost effective. We designed, despite not using it, with the following parameters: that one CO2 sensor with temperature and humidity – the Senseair K33 EGL/BGL – would’ve been using UART(the microcontroller) and the other I2C for communication (the sensor), the voltage range for the voltage supply is slightly different, and how long we must wait after “waking” the device to read data. So the design still works for both, instead of using the CO2 sensor however, we drop in replaced it with the same temperature sensor as on the main unit.

For our main unit we used the Senseair K30 model CO2 sensor. We designed communication of the chip with universal asynchronous receiver/transmitted (UART) for the communication to the Microchip PIC24FJ128GA010 main microcontroller. UART offers the ability for chips with different clock cycles to communicate with each other chip without having to take into account “padding” or added data simply to maintain a connection between the devices. Our Microchip PIC24FJ128GA010 has two UART ports available. Since we are using one UART port for communications to the ARM microprocessor the other will be used for our CO2 sensor. I2C was used for the temperature and humidity sensor. Powering the device is designed for a 24V mains power which has been from which we designed it to then be stepped down to 5V, the range for supply voltage is 4.5 to 12VDC, we were well within within the 5-12VDC recommended range in that instance. This is supplied to the G+ connector on the Senseair K30. A ground connector is supplied to the G0 pin. For the UART connections, the TxD port and RxD ports are used. The ports can be seen in the highlighted region of the schematic in figure 35. The PCB design of the Microchip PIC24FJ128GA010 controller has been modified to include a connection to CO2 Sensor via UART. The resigned main board PCB can be seen in the figure 34 directly below. So for interfacing between the microcontroller and the sensors for both UART and I2C we used 3 I/O pins total. One is the VDCC a 3.3-5V signal that when low will cause the sensors to start taking measurements, and two communication pins, one for transmitting data and the other for receiving data. For UART on the CO2 Sensor these pins are labeled TxD for transmitting data and RxD for receiving data. On the microcontroller they are U1TX for transmitting of data and U1RX for receiving data. The main microcontroller the Microchip PIC24FJ128GA010 has one additional UART port and thus also has U2TX and U2RX. For I2C on the CO2 sensor the pins are labeled, I2C\_SCL which is for the clock and I2C\_SDA for the data. On the PIC microcontroller these pins are designated by SCL1 and SDA1 for the clock and data respectfully.

Despite being limited by the availability of the easier to use UART ports on secondary microcontroller, I2C offers bidirectional lines and since the amount of data we are transmitting is small there was more than adequate for our design needs and it is cost efficient. Microcontrollers with more UART ports typically cost more money than simply I2C. Figure 36 below shows the schematic with the connections to the microcontroller.



**Figure 36 Schematic Design of PIC24 with CO2 UART CONNECTION**  
Note: WiFi connections removed – denoted by parenthesis.

The resistors labels as well as their typical, minimum, maximum and tolerance values are described below in shown in Table 5. Our microcontroller the Microchip PIC24FJ128GA010 has internal pull-up resistors so when connect the I2C\_SCL and I2C\_SDA lines we simply need to enable them on chip to be able to connect to the I2C port. Also it is important to note that the sensors run at 3.3V for the VDCC lines but are tolerant of up to 5V logical levels, which is a large tolerance for error of the microcontroller and enables us to use either the 5V or 3.3 digital out of the Microchip PIC24FJ128GA010. We choose 3.3V supply to the microcontroller to power it. We then decided for the CO2 and temperature sensors to be powered via the 5V power supply.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Resistor** | **Minimum** | **Typical** | **Max** | **Tolerance** |
| Rp\_scl (I2C SCL Pull-up) | - | 56 kOhms | - | 5% tolerance |
| Rp\_sda (I2C SDA Pull-up | - | 56 kOhms | - | 5% tolerance |
| Rs\_scl (I2C SCL Series Resistor) | - | 56 kOhms | - | 5% tolerance |
| Rs\_sda (I2C SDA Series Resistor) | - | 56 kOhms | - | 5% tolerance |
| SCL processor internal pull-up | 4 kOhms | 5.6 kOhms | 8 kOhms | - |
| SDA processor internal pull-up | 4 kOhms | 5.6 kOhms | 8 kOhms |

**Table 5 Resistor values of the K20/K21/K22/K30**

When the DVCC line goes low it triggers a measurement. We needed to delay the reading by about 15 seconds in order to give the chip enough time to properly measure and write the value to RAM. Then we read the value from RAM and store that in a register in the microcontroller which is then transmitted via XBee to the display board. Essentially are operating the Microcontroller as a master in a single master environment. In such a configuration the K30 sensor will never initiate communication. Table 6 shows a basic outline of such a sequence of events of the communication via I2C. In our design we will after step 2 we will initiate a 15 second delay for the device to make a reading. The conditions for our protocol such as the values received for ACK or NACK conditions will be determined at the time of programming the device.

|  |
| --- |
| **Communication to a Slave via I2C in a Single master environment** |
| 1. Assert a Start condition on SDA1 and SCL1.  2. Send the I2C device address byte to the slave with a write indication.  3. Wait for and verify an Acknowledge from the slave.  4. Send the first data byte (sometimes known as the command) to the slave.  5. Wait for and verify an Acknowledge from the slave.  6. Send the serial memory address low byte to the slave.  7. Repeat steps 4 and 5 until all data bytes are sent.  8. Assert a Repeated Start condition on SDA1 and SCL1.  9. Send the device address byte to the slave with a read indication.  10. Wait for and verify an Acknowledge from the slave.  11. Enable master reception to receive serial memory data.  12. Generate an ACK or NACK condition at the end of a received byte of data.  13. Generate a Stop condition on SDA1 and SCL1. |

## 3.4 LCD Screen

The LCD touch screen panel we used in our design is developed by Xenarc Technologies Corporation. We use it as the display for the system's user interface for the efficient HVAC control and feedback system. It is connected via HDMI and USB (for touch inputs) to the ARM microcontroller. Since this is one of the most expensive components of the design, we chose this screen based on the fact that it provides excellent display resolution and good brightness for its size. The display is how the user knows the system we implemented and as such we wanted to make sure it was big enough to use, displayed images with proper color, contrast and brightness and was responsive to the user’s touch. The sponsors' desire was to keep the idea of a large, touch screen user interface display area as in the first version of the system while keeping the cost down to both stay within our budget and keep the overall cost of the overall system as low as possible. We chose to purchase a 7 inch Thin Film Transistor (TFT) LCD Touch screen Monitor with both High Definition Multimedia Interface (HDMI)/Digital Visual Interface (DVI) inputs, and others however we used the HDMI interface.

The LCD touch screen can be utilized with either the user’s finger or a touch pen to provide input into various states of our HVAC control system. On the main home screen of the LCD screen the user is able to set the desired temperature and relative humidity for in cooling or heating modes by tapping on the radio buttons provided, view the current temperature and relative humidity inside the structure that it is housed, view the current temperature and relative humidity outside where the remote sensor unit is located, and choose between several settings that range from maximum savings to maximum comfort which adjusts the system's tolerance for temperature and relative humidity variation between set points. The model number of the Xenarc touch screen chosen is 700TSH. Table 6 below describes the features associated with the LCD.

|  |  |  |
| --- | --- | --- |
| **LCD Type** | **LCD Model** | **Backlight Type** |
| Automotive Grade 650NIT | TFT | Twin-tube High Brightness CCFL |

**Table 6 Important descriptors of the Xenarc LCD panel**

The key feature of the panel is the 5-Wire Touch screen. This type of display is the most durable resistive analog that has drift-free operation even when exposed to temperature fluctuation. It's approximated lifespan is 35,000,000 touches versus 1,000,000 touches from 4-wire resistive touch screen displays. Increased optical quality compared to most small scale LCDs allows for high brightness and a sharper display. With greater touch point density, faster response time and increased speed, the 7" Xenarc Touch screen monitor is capable of providing above adequate visualization to this system. The LCD touch screen contains high quality resolution and have at least 16 bit color.

The physical specifications of the LCD touch screen matches the basic requirements requested by the sponsors and provides an optimal essential function for the system's success. Since this is the only portion of the system that a potential customer would interact with, the LCD panel was chosen to be as pleasing to work with and attractive as possible. The physical specifications for the Xenarc 7 inch LCD are shown in Table 7 below.

|  |  |  |
| --- | --- | --- |
| **Item** | **Specifications** | **Unit** |
| Screen Size | 7 | inches |
| Native Resolution | 800(H) x 480 (V) WVGA | pixels |
| Supported Resolution | 640 x 480 ~ 1600 x 1200 | pixels |
| Dot Resolution | 2400 x 480 = 1,152,000 | dots |
| Contrast Ratio | 300:1 | (unitless) |
| Response Time | 6 | milliseconds |
| Viewing Angle | 140° Horizontal, 80° Vertical | degrees |
| Touch Screen Interface | USB/RS232 | (not applicable) |

**Table 7 Specifications for 7 inch Xenarc LCD Panel**

### 3.4.1 Environmental Conditions for Xenarc LCD Monitor

The resistive touch screen monitor purchased has a few environmental conditions that needed to be kept in mind. They include storage temperature and operating ambient temperature. The operating temperature can withstand a minimum value of -22° Fahrenheit (-30° Celsius) and a maximum value of 185° Fahrenheit (85° Celsius). For the storage temperature's parameter it can be held at the same temperature range as the operating temperature. Table 8 describes the environmental conditions that must be followed when dealing with the Xenarc 7” LCD with touch screen interface.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Item** | **Symbol** | **Minimum** | **Maximum** | **Unit** |
| **Operating Temperature** | TST | -30 | 85 | °C |
| **Storage Temperature** | TOP | -30 | 85 | °C |

**Table 8 Environmental conditions**

### 3.4.2 Electrical Characteristics

The most important specifications that we must follow when dealing with the Xenarc 7” LCD screen are the electrical characteristics. These characteristics include but are not limited to, digital power supply voltage, power supply current, input high threshold voltage, input low threshold voltage, and power consumption. Table 9 shown below describes how much power, and voltage that will be needed to operate the 7” Xenarc LCD Monitor. This was important when powering the device. Unfortunately due to a change in the specific model of the display we were using to a newer model with better specs and cheaper price after we designed the power we were unable to power the LCD screen without the included external adapter. We have updated the design schematics to be able to do this in future versions.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Minimum Value | Maximum Value | Unit |
| Power Supply Voltage | DC 12 | DC 12 | Volts |
| Power Consumption | <25 |  | Watts |
| Operating Voltage Range | DC 10 | 24 | Volts |

**Table 9 Xenarc Specs**

## 3.5 ARM Microprocessor Board

The ARM microprocessor is what helps to drive the Graphical User interface for the LCD display. It provides the software developer with a simple way to develop a user interface on the Linux operating system on the screen for to satisfy his or her needs without having to do a great deal of low level graphical programming which would increase the chances of system errors and exceptions. Our time would’ve been heavily spent implementing those features had we continued the design using the PIC microcontroller. The ARM is a part on the Pandaboard which is essentially the connector piece between the microcontroller and the LCD interface. The PIC microcontroller would not have been capable of producing the information and endure the constant interaction with the user and communication with the system modules because it does not have enough processing power.

The board is able to boot a working version of Linux onto the screen – in our case Ubuntu - and allow for applications written in java to run on the system. That was not possible with the pervious iteration. With the use of a modern operating system environment, users have a more dynamic experience then a static one with the bitmap constructed layout in the first version of the HVAC Feedback system. The high level board is required to be programmed separately from the microcontroller and utilize a separate procedure to accomplish this portion of the programming. The ARM microprocessor is even more essential to the overall system than the LCD screen itself. Although the LCD screen is what the user interacts with and is the only part of the system to be seen by the user, the high level board is responsible for making sure the LCD screen performs the tasks expected of it by the user and responds accurately to the user input and acts as the web server for remote internet access.

The Panda board has many features that will satisfy the design requirements of our group. The features of the Panda board are listed below in Table 10.

|  |  |
| --- | --- |
| Features of the Beagle board (high level controller) | |
| 1-GHz super scalar ARM CortexTM-A8 | JTAG |
| 512-MB LPDDR RAM | High-Capacity microSD slot and 4-GB microSD card |
| High-speed USB 2.0 OTG port optionally powers the board | HD video capable C64x+TM DSP core |
| On-board four-port high speed USB 2.0 hub with 10/100 Ethernet | On-board Ethernet and four SB 2.0 ports that support low, full and high speeds |
| DVI-D port | Open GL ES 2.0 capable 2D/3D graphics accelerator |
| S-video port | Boots from external media, such as microSD slot, serial port or USB port |

**Table 10 HLCU features**

### 3.5.1 Electrical Characteristics

The electrical characteristics of the Pandaboard are important and must be taken into consideration for the overall design and stability of the Efficient HVAC Feedback System. These specifications were considered throughout the development of the system and in the final design of the system. The board draws about 5 V DC and max 4A.

In order for the Pandaboard to operate it requires connections for power, communications, panel, and inverter. A DC power supply is used to provide the system with sufficient power so that the board may function efficiently. Supplies that provide additional current than what is specified can be used if additional current is needed for add on accessories.

The amount specified is equal to that supplied by a USB port. A voltage of 1.8V is driven through the board in operation to various components. Table 11 shows the DC power supply specifications.

|  |  |  |
| --- | --- | --- |
| Specification | Requirement | Unit |
| Voltage | 5.0 | V |
| Current | 4 | A |
| Connector | 2.1mm x 5.5mm Tip positive polarity |  |

**Table 11 DC Power Specs**

The Pandaboard supports RS-232 for serial communication via UART3 is provided by DB9 connector on the Beagle Board for access to an onboard RS232 transceiver. A USB to Serial cable can be plugged directly into the and also used as a RS-232 port. A standard male to female straight DB9 cable may also be used. Below is an illustration of this in Figure 37.



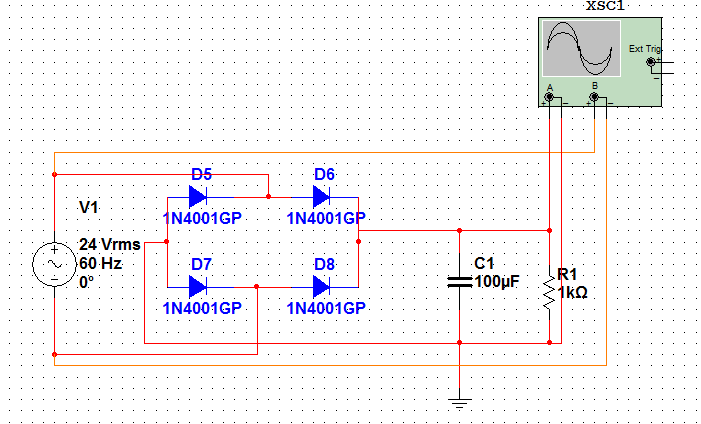
**Figure 37 HLCU Communication Setup**

## 3.6 Power System

The power system of the original HVAC system proved to be challenging and a cause of a lot of issues. Parts with insignificant tolerances and specifications -specifically the amount of current being drawn from the linear regulators and also likely was the trace width - lead to breakdowns and overheating. The problems were found and we designed using switching regulators instead of linear regulators and other design changes to solve these problems.

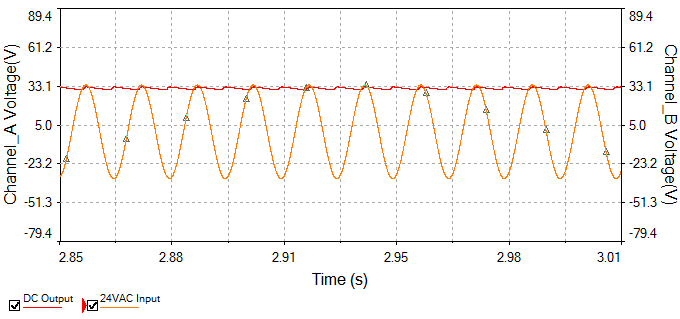
### 3.6.1 Main Unit

The main unit is powered by a 24VAC input as is common in all household HVAC controls. Current pull was a slight issue in design as we are powering a significant sized LED screen, rated at about 1A itself, the Pandaboard is 4A. The initial design assumed 2Amps to be more than sufficient to power the micro-controller and other sensors. The ARM processor, and LED screen combined max are 5Amps. The device is hooked up using a full wave rectifier very much similar to Figure 38.

****

**Figure 38 24VAC Full Wave Rectifier**

Running a full wave rectifier through the software simulation package Multisim we were able to determine our theoretical output DC voltage returned us the following graph in Figure 39. The yellow line is the AC input and the red line is the DC output after going through the rectifier.



**Figure 39 Rectifier Output**

This rectified DC Voltage output is approximately 33.1 VDC. In practice however using a 120Volt to 24Volt transformer without load our output voltage was 38 volts DC after rectification. This was due to the supply voltage form the outlet being higher than theoretically assumed. Furthermore the lack of load affects the output voltage. With load the voltage before rectification is 27Volts. This was more than within our design considerations however. We then took this and regulated to 5V and 3.3V for our circuit. Due to some of the loads need high amp outputs switching regulators were used. National Semiconductors LM2679 Switching regulators fit the specifications having both a 5 Volt and 3.3 Volt model. Each of these required several capacitors, inductor, and a Schottky diode to function correctly. The main control boards microcontroller and other smaller IC’s were powered by a linear regulator connected to a 5 Volt switching regulator The linear regulator used on our main board is the LD1117 sold from SparkFun.

The original diodes used were rated for 1A. Instead of using these, triple the current can be allowed using Motorola's 1N5400 available at futurlec.com. The capacitor in the original device was also severely lacking. The capacitor will be increased to 100uF to help limit ripple and provide a nice smooth DC supply. Further, the voltage rating should be high enough. DigiKey's part P1191-ND is a 100uF capacitor rated for 35V and is bipolar.

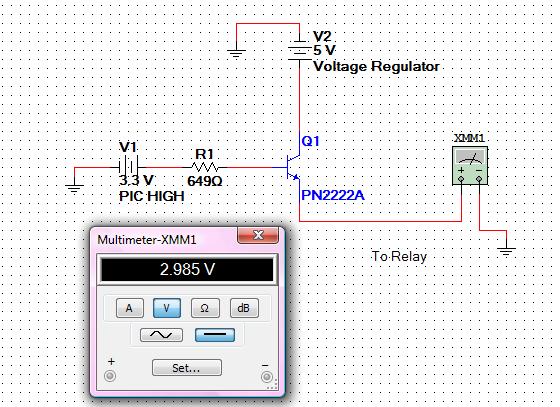
### 3.7.2 Secondary Unit

The secondary remote unit is powered by 2 AA batteries put into an IC output at 5V that was rectified in order to power the microcontroller, temp sensors, and XBee chip. The temperature sensor chosen operates at a supply voltage of 3.3V. The secondary microcontroller operates between 1.8V – 3.6V. The XBee chip operates 2.1V – 3.6V so a steady voltage of 3.3V will be perfect for our secondary system. OLIMEX makes a small solar cell attached to a single AA batter that can provide 3.3V that was originally intended to power a TIMSP430 processor but is comparable to our secondary remote unit. SparkFun electronics sells it as SKU: DEV-08251. When the solar panel doesn’t provide enough power, the batteries can be used.

## 3.7 Relays

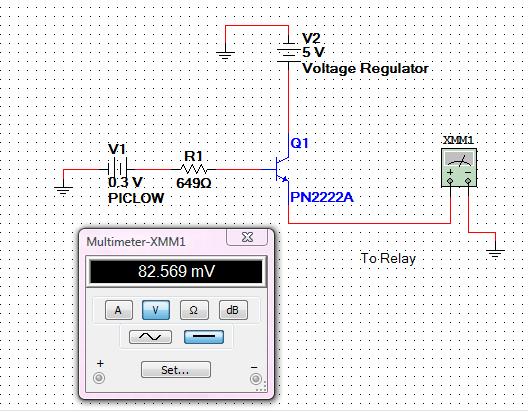
Though the microcontroller controls all the logic and has the pins to activate all the components of the system, the necessary activation voltage is not adequate coming from a micro-controller. In order to solve this problem, the output of the micro-controller will be outputted to controlling relays. Either one or more relays must be able to be turned on or off at different times. A mux would not have been adequate to control all the relays possible so a tie in to the microcontroller is the only way it can be done feasible. The relays will be used to control the air conditioner units, heaters, fans, de-humidifiers, mood scents, dampers, and the natural air system. All these are powered using 24VAC, which is common wiring in the house and buildings for HVAC components is what we designed our power system around in the system. Having the 24VAC power on the air conditioner, heater, fan are common protocol in houses, and because of this there are many options for 24VAC dampers too. Seeing as the microcontroller has a DC output of 3.3VDC, it is unlikely this voltage could be used to send signal for the distances necessary in a house. Further this signal wouldn’t provide the strength necessary to also activate the device.

We use reverse logic for the control of the relays. An output of low from the PIC provides 200-300mV and the high of 3V. Then this output then goes right through to an NPN transistor. The reverse logic in the microcontroller is the high coming out of the main controller will keep the relay in the open position. When the relay is open, the typical device attached to the HVAC system will not be receiving its activation voltage of 24VAC, making that unit off. So a high logic from the microcontroller turns off a portion of the system. This is shown in the Figure 40 below when the relay will be open switch.



**Figure 40 Relay open**

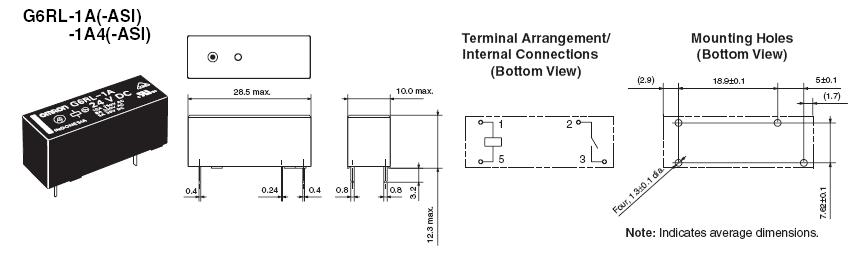
When the relay is to be closed, a non 3V signal is to be sent to the relay. This is the need of reverse logic. Now the micro-controller will go low, allowing only 200mV or so through the NPN transistor to the relay closing the switch. When the relays switch is closed, the Voltage is applied on the other pin allowing the device at the other in to turn on. All the ports will be initialized to keep the relays off, and the code inside of the microcontroller will then decide the logic between turning on and off the relays. This code could have easily been implemented on the ARM control unit and the signal sent to turn on and off sent to the microcontroller, causing communication to the relay being the only function of the microcontroller when determining the proper logic of the HVAC system. We determined that instead, only user interface information, such as what temperature to turn on air conditioner, is sent from the ARM display unit, and the main microcontroller will contain the math to determine logically weather to turn on or off each specified relays. This holds true for the mood scents as well, as the information of what the user wants may be entered into the ARM, but that information is sent to the microcontroller and it will make the logical decision of when and how long to turn on the mood scents. The relay going closed and allowing these devices to turn on are shown below in Figure 41.



**Figure 41 Relay Closed**

The purpose of the NPN transistor is to provide protection to the PIC. When a relay is on there is a large amount of current going through it creating a magnetic field. When the relay’s is switched back off, the sudden collapse of the magnetic field inside of the relay produces a brief high voltage cross the whole circuit. It is important to provide some protection by some kind of flow switch. Commonly diodes are used, or in this case an NPN transistor controls the flow of current, making sure it is only going towards the relays and not from the relays.

The relay chosen for our system will be Omron’s G6RL. This has a working voltage of 3-3.3V. Pin 1 will have 3.3 Volts tied to it straight from the voltage regulator coming from the power supply. Pin 5 receives the output from the PIC as seen above through the NPN transistor. Pin 2 will be connected to the 24VAC voltage source before it goes through the full wave rectifier, and pin 3 is going to go out to the system desired to turn on. A diagram of the relay to be used is below with its holes in Figure 42.

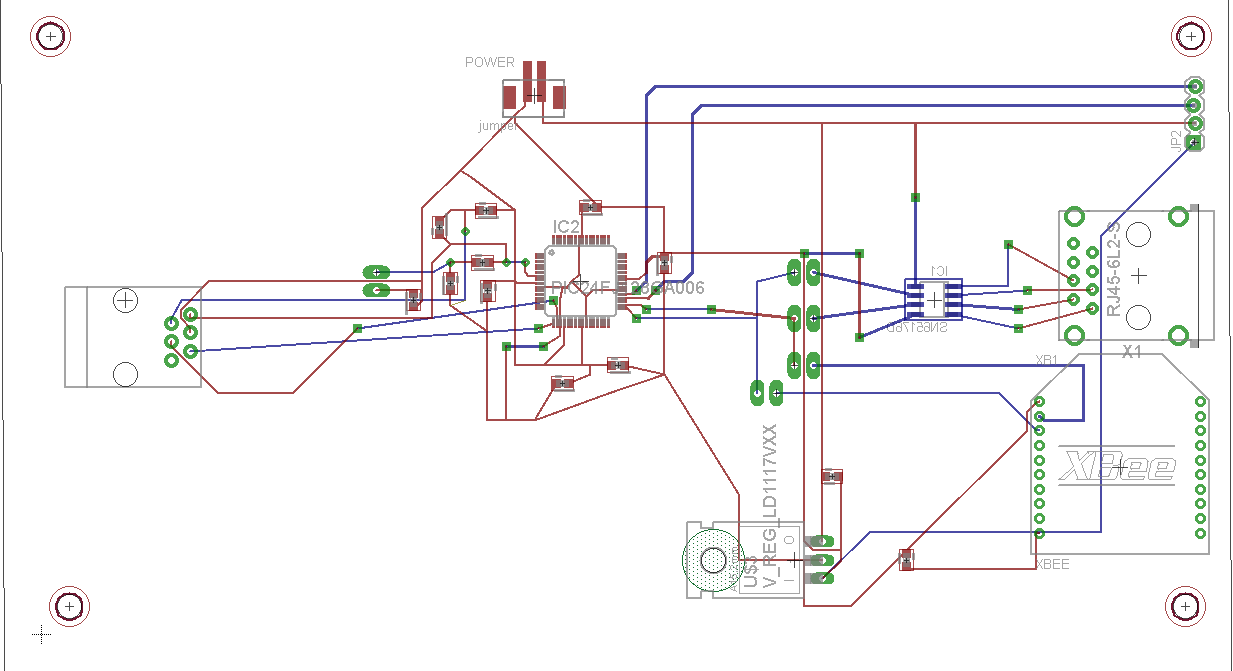


**Figure 42 Relay Overview**

**(Permission Granted From Omron)**

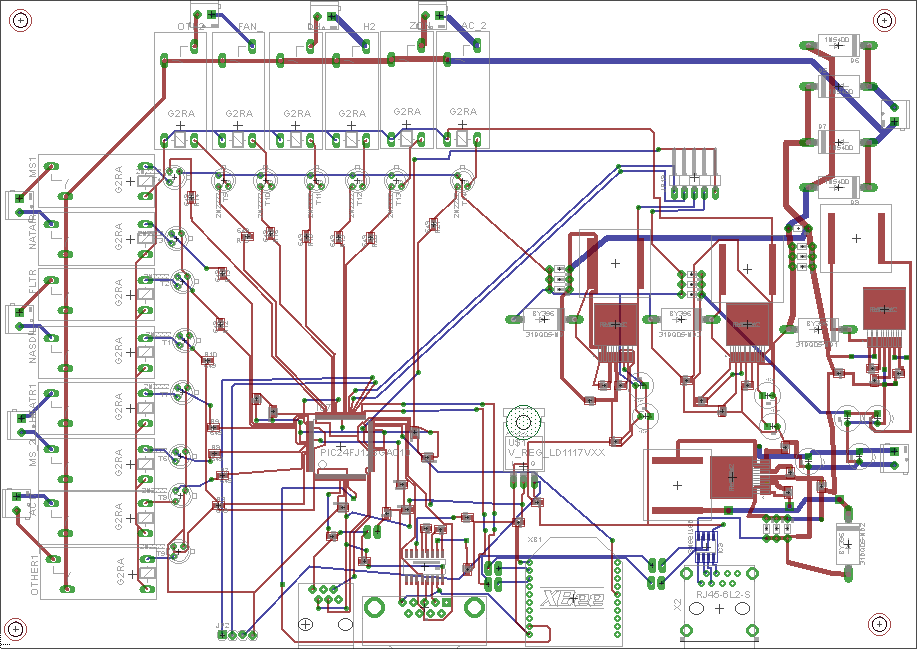
# 3.8 PCB

Our PCB’s were designed using CADSOFT’s Eagle. After first locating or creating all of the parts in the schematic editor, the program was switched to the boards. The secondary unit was the first one that was attempted as it was much less complex and provided a good learning tool. The board proved to be mildly challenging to route all of the wires without shorts. We then ran several software checks in Eagle to make sure none of our wires or vias were too close to short or provide interference. The final PCB schematic is seen below in Figure 43.



**Figure 43 Secondary PCB**

# The main board provided a much greater challenge. The board contained a much greater list of parts. Further, in contrast to the secondary board, not all of the parts were located in the Eagle library. Parts had to be made in the Eagle library editor. Afterwards, the significant increase in parts caused increased difficulty in routing the wires. These wires also had to have their depth and width due to increased current requirements on the main board as discussed in the power section. Below in Figure 44 is the final PCB of the main board.



**Figure 44 Main PCB**

# 4 Prototype

Throughout this phase, we have put tremendous dedication to the operation and cosmetic outlook of the prototype. In order to be fully aware of the current state, we have included 14 LEDs to indicate the components that are currently activated. Further, we have integrated the entire power supply to allow the unit to power all devices on the prototype such as would be used in actual use. The software is handled so as the user interaction was the focal point. For presentation purposes, we have designed the system to be enclosed by a transparent case so the demonstration can display the entirety of the unit in one device. Where these parts and how we implemented this prototype will be discussed in more detail in this section. Figure 45 in the following section depicts the device in prototype phase.

## 4.1 Vendors

The vendors chosen to provide the system required parts served a special role and purpose in the development of the HVAC control system and also have a direct association to our sponsors. All of these companies provided us essential components for our wireless microcontrollers, development board, LCD touch screen display, and wireless interfacing to communicate between the microchip and the router. We researched these companies amongst others and thoroughly thought that they would provide us a great opportunity to get excellent products that focus on low power consumption (in BTUs), energy efficiency, and eco-friendly air quality. They also allowed for satisfactory comfort to the consumer, and to supply heating and cooling for the commercial or residential unit according to their consumers preferred temperature and humidity settings. Each vendor's area of expertise in the overall construction of the entire system's microelectronics is described in further detail.



**Figure 45 Prototype Unit**

### 4.1.1 Xenarc Technologies Corporation

Xenarc Technologies Corporation is a corporation located in Santa Ana, California that specialized in VGA and DVI input TFT touchscreen LCD monitors of smaller size to fit today's in-vehicle and system integration applications. Since 1995 they have been manufacturing LCD monitors. The qualities that Xenarc Technologies Corporation offers are product durability, great display quality, and adequate battery operation. This company was appealing to us to use because we ordered a 7 inch LCD Display Module for our HVAC control system and their LCD screen fit our design requirments.

### 4.1.2 Microchip

Microchip Technology is a company whose primary concentration is in the semiconductor industry. The company's base of operations is located in Chandler, Arizona consisting of approximately 5,000 employees where they produce electronic appliances such as microcontrollers, memory and analog semiconductors. Memory products include serial EEPROMS and serial SRAM as well as other analog devices. The company was basically founded by Steve Sanghi, J Bjornholt, and Ganesh Moorthyin 1989. Even though it was originally founded in 1989, they actually existed in 1987 when General Instrument split off their microelectronic department into their privately owned company. We were interested in their products because of the wireless interfacing being involved between the control panel and the router. Our group ordered wireless network devices on their main website for the integrated circuit chip such as the Zigbee MRF24WBOMA and the Wi-Fi for the development board inside the HVAC control panel. Below is the schematic of the Microchips connections and its programming pins.

### 4.1.3 Sensirion

Sensirion is an engineering firm that is located in the municipality in the Canton of Zürich, Switzerland. This company consists of more than 180 employees that manufacturer sensor parts primarily for OEM applications. Sensirion was created in 1998 by Felix Mayer and Moritz Lechner. The products developed at Sensirion are applied in the medical field, automotive industry, and HVAC systems. The technology for the sensors integrates CMOS circuitry. Noticing that these products are used in the HVAC systems, we ordered humidity and temperature sensors from their company's website. This company played a big role for our project because of the need for our HVAC control panel to read the inside and outside temperature and humidity settings. When looking at which part would best work in our system, we had to make sure that these sensors would meet the conditions that the sponsors wanted. In order to meet the conditions, the component that would do that is the SHT21 temperature / relative humidity sensor.

### 4.1.4 DigiKey

DigiKey is the 5th largest electronic component distributor in the United States. They are committed to a fill rate of over 95% when typical industry average in this section is below 50%. DigiKey offers fast shipping options for our parts as well as customer support. Several of our members have had positive experiences with DigiKey, and we will order several of our miscellaneous parts such as our resistors and voltage regulators from Digikey.

### 4.1.5 Senseair

Senseair is the company that makes our CO2 sensors for the project. We consulted with them during the design process and were happy that the outcome of such collaboration resulted in a well-planned out design. Their customer service was top notch and even offered engineering samples for our first prototype with the CO2 sensors. We have since gladly accepted the offer of discounted engineering samples as we are a student project, and the sensor we received worked great in our design. Very professional and customer oriented. Simply put look to them first for CO2 Sensors.

### 4.1.6 CO2Sensors.com

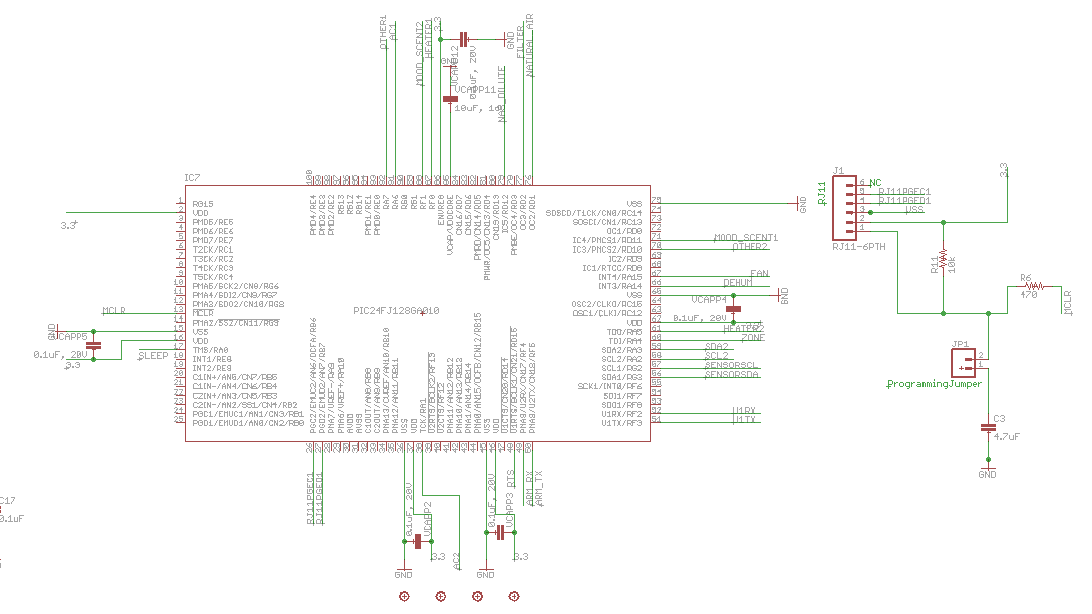
CO2sensors.com (CO2s) offers a wide variety of CO2 sensors on their site. Since we are only getting engineering samples from Senseair, we still do need a UART to USB cable to program and calibrate the sensor for use. This cable could’ve been useful for diagnostic purposes as we could’ve use Senseair’s software to ensure the device is working properly independent of the circuit. We ended up not using it however as it wasn’t nessicary. CO2sensors.com also offers many user guides such as basic I2C programming for the Senseair K30 for the Arduino microcontroller platform which was a great reference to our design. Their customer service has thus far been very response and should there be a need for more CO2 sensors for a future prototype, we highly suggest this company for ordering the Senseair CO2 sensors.

## 4.2 Schematics

In order to show a proper understanding of our system, all of the components researched necessary to implement in our system must work cohesively. It is important that all sections of our system be able to communicate with each other when necessary and to not cause power issues. The following schematics show the basic connections of our system. This section includes a printed circuit board section when the development boards have been adequately tested and debugged, allowing us to completely implement our system on a printed circuit board.

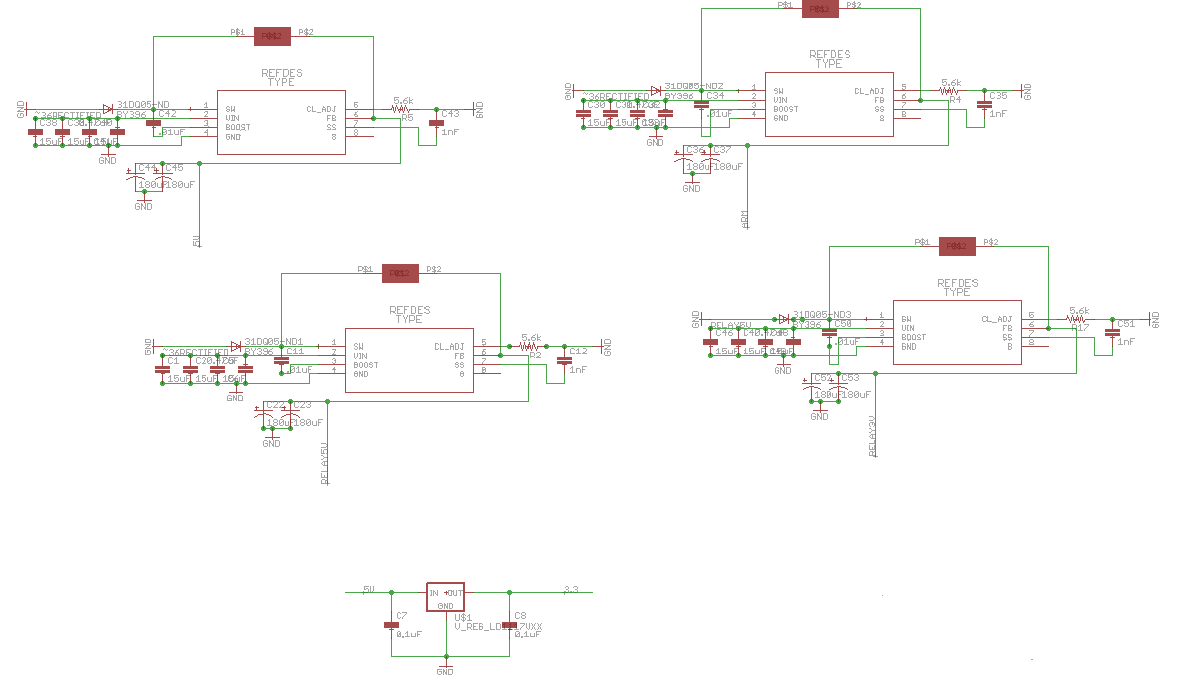
### 4.2.1 Main Master Control Unit

All of the systems we've previously researched and implemented into our design need to work together. Finding the correct way to connect and communicate them was the majority of our time in senior design two. The following figure is the final schematic of our main control unit. Starting at the bottom is the 24VAC input. From there on the left is the outputs from the arrays, which is tied directly to the 24VAC output. These relays are connected to the PIC micro-controller with a NPN transistor between. We use an NPN because it is running on reverse logic at this time. 24VAC input is put through a full wave rectifier to bring out a 34VDC that must then be regulated down to the usable voltages of our system. There are two 5VDC regulators. The purpose in using two is to require less current across each one, effectively cutting it in half. The ARM micro-controller and LCD has one of these regulators completely devoted to them. The other 5V regulator is to be used on the main control board. The majority of our parts were powered through 3.3VDC though so another regulator is put underneath the two 5V regulators. The 3.3VDC regulator is used to power the PIC, Xbee, CO2/Temp/Humidity Sensor, and powering the relay controls. An RJ11 port is going to be used to program the chip while an RJ45 was placed on the board to allow wired connectivity with the secondary control unit. Figure 46 below is the schematic of the main microcontroller and programming pins:



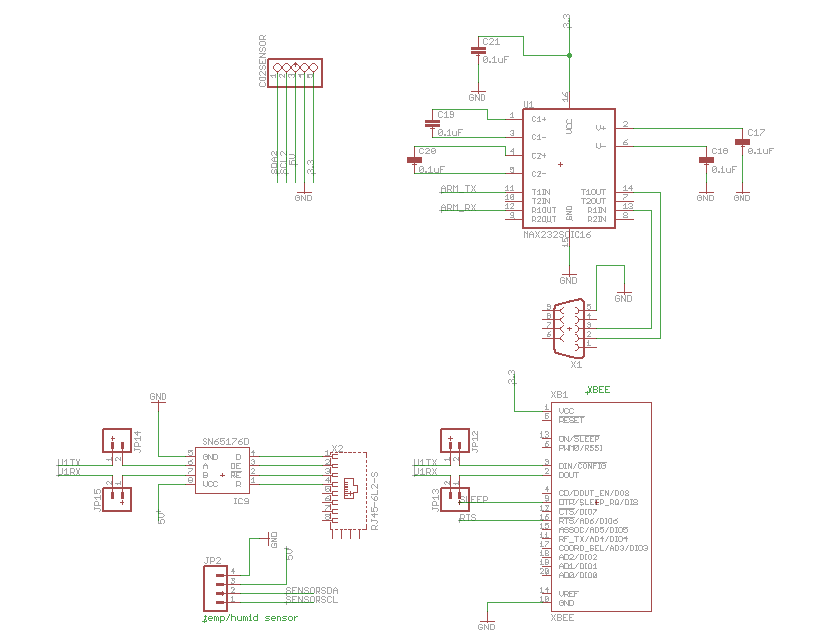
**Figure 46 Main Controller Schematic**

The power system connections were made on one side to keep interference from the inductors to a minimum. The switching regulators and inductors had to be hand made in Eagle in order to get the footprints correct. There are four switching regulators on the board with their accompanying capacitors, inductors, resistors, and schottky diodes. Further there is one 3.3 Volt regulator to to connect to the main microcontroller and sensors. Below in Figure 47 is the schematic of the power systems.



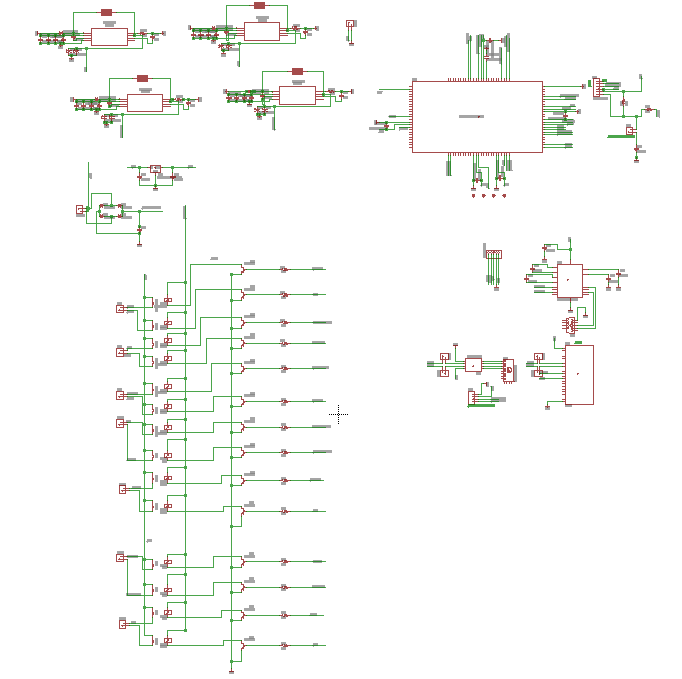
**Figure 47 Power Schematic**

The CO2 and Temperature and Humidity sensors are connected to the main microcontroller using I2C interfacing. The XBee and RS232 are connected to the UART ports on the main microcontroller. The schematics of these are seen below in Figure 48.



**Figure 48 Sensor Connections**

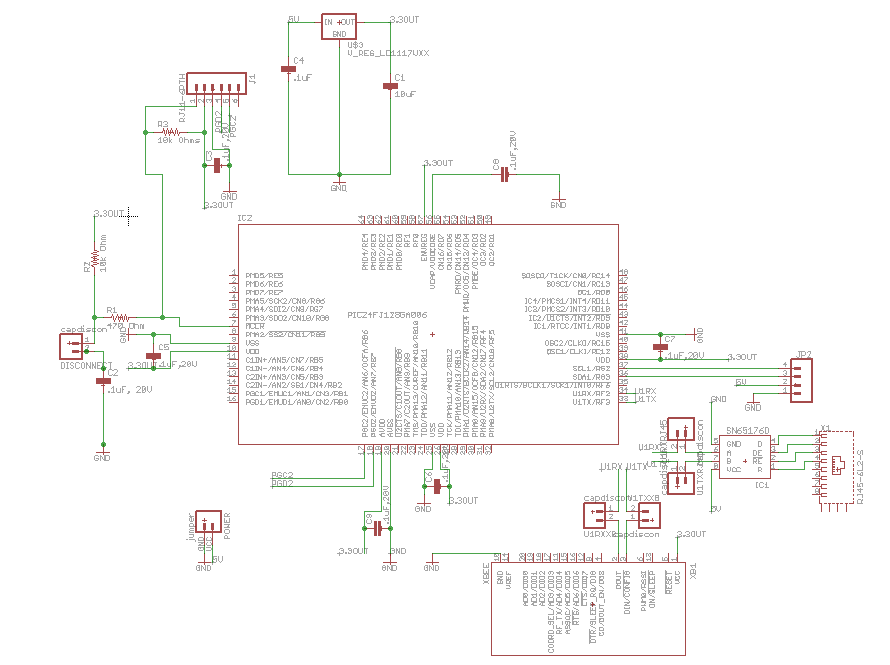
### The entire schematic is large and almost incapable of reading in its entirety, which is why the above sections were partitioned. Below in Figure 49 is the entire schematic with the added relays.



**Figure 49 Main Schematic**

### 4.2.2. Secondary Unit Schematics

The secondary microcontroller unit is separated from the main unit so it needs its own system. The secondary microcontroller is a much smaller PCB board. It has two means of communication. The most commonly used one is the Xbee chip to communicate over radio frequency to the main microcontroller. In circumstances where the walls are too thick or other issues arise making wireless communication impossible or unreliable and RJ45 (wired) input is available to directly connect the secondary and main microcontrollers over a twisted pair Ethernet connection. The RJ11 phone port on the system is used for programming the microcontroller. The system is powered mainly by the battery that is to be charged by a small solar panel. Lastly, the entire reason of the unit is to gather temperature and humidity readings. There is a sensor attached to the PIC microcontroller that gathers this data and send it to the main microcontroller in between sleep cycles. Figure 50 below is the basic new schematic of the secondary outdoor unit.



**Figure 50 Secondary Schematic**

## 4.3 Main Control Unit

The design of the main control unit is powered by the 24 V AC common wire that is installed in the building during the initial construction project. The advantage of this design is that the consumer doesn’t have the hassle of having to recharge batteries for the thermostat or have to put in any additional wires. This is also a cost savings for the sponsor. Of course, a future upgrade may be to add in batteries as a back-up power source in case there is power outage in the house. Having batteries to power the device would call for a bit more conservative approach since the parts would have to be low power in order to sustain a long product working time on one charge.

The main control unit houses several components of the overall design. The components located in the main control unit are the SHT21 Sensor, the PIC24FJ128GA010 main microcontroller, the Xbee 802.15 Transceiver, CO2 Sensor, and relays. The main control unit of the HVAC system also includes the LCD screen and the Pandaboard which is be the main hub for all of our operating system needs. Our board uses Ubuntu Linux 11.04 on the Pandboard which allows us to create the sophisticated GUI. On this board, we have a WIFI module which connects to the internet for various needs such as live weather report, online configuration and so forth. The GUI has the ability for the user to input a schedule of what temperatures they would like to set for specific days of the week and also the times of scent modes. Having this program be responsive was very important and we felt we met this requirement.

All of our components are within a reasonable operating voltage range which would require us to convert the 24 V AC from the wall down to voltage of 3.3 V and 5V. Our design includes a full wave rectifier followed by voltage regulators. Making a decision on the voltage regulators included choosing the right regulator that can handle the 32 V max and give us the sufficient output current the load is going to require. The Xbee chip has an operating voltage range of 2.4 V – 3.6 V with a typical voltage of 3.3 V. The main microcontroller has an operating voltage range of 3 V - 3.6 V. The temperature and humidity sensor has an operating voltage range of 2.1 V – 3.6 V with a typical voltage of 3 V. The CO2 needs a typical operating voltage of 5V. These voltages are show here in Table 12 below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Min Operating**  **Voltage (V)** | **Typical Operating Voltage (V)** | **Max Operating Voltage (V)** |
| Main Microcontroller | 3 | N/A | 3.6 |
| Xbee wireless chip | 2.4 | 3.3 | 3.6 |
| CO2 Sensor | 5.0 | 9.0 | 14.0 |
| Temperature and Relative Humidity sensor | 2.1 | 3 | 3.6 |

**Table 12 Accessory Voltage**

In order to transform the input voltage from 24 V AC to 3.3 V DC and 5V DC, several steps needed to be taken in order to achieve this. First the signal needs to be converted from AC to DC by the method of a full wave rectifier. This method is also called bridge rectification and is accomplished through the implementation of four diodes. Four diodes were used, two at each time accounting the positive side of the input sinusoidal voltage signal and the negative side of the input sinusoidal voltage signal. The bridge is an arrangement of four diodes in a diamond configuration that no matter the polarity of the input, the polarity of output is the same. In essence, when the AC signal goes below zero, the output from the bridge rectifier would be that same pattern but reflected in the positive domain. Therefore, upon advancement through the bridge rectifier, we have a signal all in the positive y plane.

At that time the signal is of one polarity but still is in a periodic form which doesn’t make it a DC signal. The signal is continuously varying or “pulsating” in magnitude from zero to its magnitude. This type of variance is not good for our parts as they require a constant voltage with a value of their threshold. If we are to put this signal in, the parts would transition through the on and off stage possibly causing damage. In order to smooth out this variance, we included a capacitor, known as reservoir or smoothing capacitor. The smoothing capacitor is dependent upon the voltage regulator implemented. We use a capacitor simply because the voltage of a capacitor cannot change instantaneously which gives us a signal that is a lot smoother. Of course, the signal after the capacitor is not perfect since ripples exist, but the variance is not as detrimental as the signal ripples around its maximum.

Once this process is done, the signal is considered close enough to true DC signal to be sent though the voltage regulator. The voltage regulator is designed to take in a threshold input voltage and produce a desired voltage. The regulator also has specifications as to the amount of current that can be drawn from it. This was critical to our design since we realized that the previous design didn’t take into account the amount of current. For the main control unit, we needed output voltages of 3.3 V, 5 V and 12 V, which required us to purchase three different voltage regulators to power our components on the main control unit. All of this is was essential to get all of our components working properly, such that the PIC microcontroller, the temperature and humidity sensor, CO2 sensor, Xbee wireless chip, LCD and ARM board all had a stable operation. We succeeded in this as our board is very stable.

## 

## 4.4 Remote Sensing Unit

Our remote sensing unit is designed to be mounted on the outside of the building in order for us to get temperature and humidity information of outside. Since there probably won’t be a power supply, we have to come up with a solution to power this secondary unit. The two options that designed around was powering via batteries or having solar panels. We choose both. Using batteries gave us a limited time of use but makes it cheap to power as no cables need to be run. It also makes it very portable in that it can be installed anywhere outside the building. The batteries are also exchangeable and cost effective. Based on the batteries that were used in version 1.0, we get at least 2 hours of sensor reading and transmit time on AA 3 V batteries. The other option that we decided to design and implement was to use solar panel in addition to the batteries. This gives the design the ability to last for a longer period of time without replacing the batteries. Compared to using just batteries, this option is more expensive but seems to be more functional. Our decision to use both gives us the best of both worlds and when considering maintenance issues and the functionality of the unit the added expense of the solar panel outweighs these disadvantages.

The components inside the remote sensing unit are the temperature/ humidity sensor, the XBee wireless chip and the secondary microcontroller. The temperature/ humidity sensor has a typical operational voltage of 3 V. The Xbee wireless chip has a typical voltage of 3.3 V while the secondary microcontroller that we have chosen has the same operating voltage. Table 13 shows a typical operating range for the components in the remote sensing unit:

|  |  |  |  |
| --- | --- | --- | --- |
| **Component** | **Min Operating Voltage (V)** | **Typical Operating Voltage (V)** | **Max Operating Voltage (V)** |
| Secondary Microcontroller | 1.8 | 3.3 | 3.6 |
| Xbee Wireless Chip | 2.8 | 3.3 | 3.4 |
| Temperature and Relative Humidity Sensor | 2.1 | 3.6 | 5 |

**Table 13 Operating Voltage Range for Components of Remote Sensing Unit**

Due to the fact that we are sourcing these voltages with the battery, the amount of current drawn by each component had to be accounted for. This is important because the battery we choose had to be capable of providing of providing the required amount of current for an acceptable period of time. For the use of the secondary unit, we should have a decent operating time such that the user won’t have to replace the batteries for an extended amount of time. The secondary microcontroller draws 200 mA max, the XBee chip draws 50 mA and the temperature and humidity sensor draws 100 mA max. This is tabled below:

|  |  |
| --- | --- |
| **Component** | **Max Current (mA)** |
| PIC microcontroller | 200 |
| Xbee Chip | 50 |
| Temperature/Humidity Sensor | 100 |

**Table 14 Current use**

The battery we chose was a 3.6 V lithium battery to power the remote sensing unit. The nominal capacity of the battery is 2.1 Ah which we have determined to be sufficient to source our components for an extended period of time. The battery is charged with a small solar panel, which provides enough excess power to the device to power it though out the night as well. The temperature and relative humidity sensor, secondary microcontroller and the Xbee wireless chip all have sleep modes which are specifically design to be low power consumption when they are not being used. The remote sensing unit is not taking readings continuously, which is why we are able to implement sleep mode. The unit is taking readings approximately once a second and then going to sleep. This cycle is in the programming of the microcontroller. Once it takes the readings, it sends the information to the main control unit, then return to sleep mode until it is time to make another reading. Using this reading technique allows us to consume less power from the batteries therefore increasing our operating time. In order to connect the battery to the PCB, we used a battery holder that was wired to the board. The battery holder has contacts that connects with the battery as well as a wire with contacts that then connects into the PCB. This method allows power transfer from the battery to our PCB. The 3.6 V is bumped up via a circuit on the battery pack to a 5V output and then is converted to 3.3 V in order to power our components on the PCB. The 3.3 volts is achieved with our 3.3 V voltage regulator which is also used on our main control unit to power our PIC microcontroller. Since the output of the battery is a DC source, there was no need to do any rectification and the output of the battery can go straight to the voltage regulator.

## 4.5 Relays

The purpose in choosing a microcontroller on the main board with so many IO pins was mostly for controlling as many devices as possible. AC1, AC2, Fans, zones, mood scents are all controlled with a 24VAC power line. In order to control these turning on without having to have this large AC Voltage and avoid DC/AC conversion, we just used relays. We added several relays for controlling the dampers for zone control and for controlling the mood scents. The relays should be able to be controlled using our microcontrollers 3V output.

Omron makes a G6RL which is designed for PCB mounting. Its switch is activated with 3VDC and the actual line can handle 10 A up to 250VAC which is more than adequate for our 24VAC signals. Relays are known to have a kickback current so NPN BJT transistors are connected between the relay and microcontroller to act as a diode. The relay is open with 3V applied to both ends and closes when there is a difference in voltage across the nodes. This required programming of reverse logic in the microcontroller to control the relays.

4.6 Demonstration

The final phase in our project's lifecycle is a demonstration of the HVAC feedback and control system. The system is required to be completely portable, self sufficient, and compact so that it can be tested within a room in front of a panel of professors fulfilling all of its expects requirements. Due to the fact the system is meant to be installed within the wall of a building, alterations were made to accommodate for this shortcoming so that the system can be properly evaluated by the panel. There is no option for us to temporarily mount the unit to the wall because the demonstration has a time limit and mounting the system in such a manner is not necessary.

### 4.6.1 High Level Control Unit and Main Control Unit

The high level control unit and the main control unit are mounted to a portable acrylic structure so that it may simulate the concept of having the system mounted to a wall within a building but also be visible for display of the design and functionality. The acrylic structure is made from light weight acrylic to keep the weight as low as possible and fastened together with screws and bolts. In terms of the LCD screen, a stand is included with the screen and we made it so it is free standing on top of the acrylic structure unit. During the course of the demonstration, the LCD screen is involved with a series of interactions to illustrate the systems functionality and feasibility. If possible, while demonstrating this system a person unfamiliar with the direct development of the system would be allowed to illustrate the simplicity of the user interface's design.

### 4.6.2 Remote Sensing Unit

The remote sensing unit is designed to be mounted outside of a building. It is designed to be powered by an independent battery source which is dedicated to the operation of the remote sensing unit only. This unit is the smallest unit among all the components in the entire system and wireless range of freedom is a factor. The remote sensing unit communicates back and forth with the main control unit wirelessly through the utilization of Xbee modules. The components of this unit are mounted on a printed circuit board (PCB). There is a temperature/relative humidity sensor, microcontroller, Xbee module and battery mounted to the board. Since this unit is designed to be positioned outside, the PCB is placed in a durable container to protect it from various harsh conditions that would compromises its functionality.

Since the unit is enclosed in this type of housing the relative humidity and temperature sensor on the board was designed for its needed exposed to airflow to accurately present the correct temperature and humidity. For this to be possible, we have designed a mechanism that would allow for the sensor to obtain adequate airflow without increasing the possibility of the PCB or sensor becoming defective due to overexposure or damaged. The simplest way to achieve this was to implement a ventilation system that allows for airflow in but keeps rain, sleet, and snow out. After various sessions of discussion we decided to place vents at the bottom of the housing which would allow for sufficient airflow and greatly reduce the possibility of precipitation infiltration.

During our project demonstration, we have planned not to have multiple air compressors, air handlers and dehumidifiers. This would be unfeasible to illustrate our system utilization of these devices in our demonstration room. We have decided to use an alternative means to demonstrate that our system is providing and that is LED bulbs. For the demonstration we have planned to use the user interface on the LCD to set a temperature for the system to maintain. Once the temperature is set we then use either a hair blower or a source of cold air to raise or lower the temperature received by the main control unit's sensor. This process represents the temperature inside the building either raising or lowering. We plan to also manipulate the temperature at the remote sensing unit to simulate the outdoor temperature. Depending on the temperature change at both locations along with user settings the system comes up with the correct course of action to maintain the user set points and take the appropriate course of action.

We have set up several different scenarios to demonstrate the wide range of situations the system can handle. For example we can have the user select the option to set “maximum energy savings” on the user interface. For this scenario we may raise the indoor temperature by only one degree. The system recognizes this is the situation and does not turn anything on because the user has told the system that there must be more than a one degree discrepancy in the set value and the actual value before it can begin cooling the building. Another scenario that we can have is that the user selection of the “maximum comfort” option and raise the inside temperature one or more degrees above the set point. Also, we plan on manipulating the outdoor temperature to be desirable for cooling the building. In this situation, the system recognizes the outdoor air is applicable for cooling the building and instead of using the compressor it simply brings the outside air into the building to bring the temperature back to the desired set point.

Unlike the first version of the HVAC feedback and control system, a demonstration covering the temperature scheduler and administrator login is planned to be demonstrated. We can show this by having the user select the "Scheduler" tab and select the desired day of the current week displayed. Within the day selected, the user sets the desired temperature for the system to achieved levels at the selected time of that day and the duration associated with this request. Once the schedule time arrives, the LEDs for the relevant appliances that are triggered in response to the temperature setting should illuminate or become enactive. Another new feature that this system version has that can be demonstrated the "Technician Login" Tab. Under this tab, the user can select the technician username and input a numerical password to reveal the "Component Enable" page where they can select what components are installed to work with our system and verify that their presence is valid.

These scenarios demonstrate the decision making capabilities of the high level control unit in conjunction with the main control unit and the wireless / remote sensing capabilities of the remote sensing unit. In addition to demonstrating the functionality of these aspects of the control system, we also demonstrate the wireless control capabilities. For this style of control, we demonstrate the mobile website for the system where users use their smart phones to log into the web site and manipulate the set points of the system over the internet instead of by using the LCD touch screen in the room.

Since access to AC units within the demonstration room is not going to be possible and we can’t integrate this system to the HVAC system already in place for the room we had to find an alternative to showing that the 24V AC control voltage is being delivered to the appropriate appliances available. In order to accomplish this alternative, we replaced the physical presence of the AC units and dehumidifier with the lighting of LED lights turning on and off to represent the various components turning on and off. With this substitution method, the LED lights are illuminated for the same duration of time that actual devices would be active for. To show the proper voltage is on the line a voltmeter is planned to be used to illustrate that the correct amount of voltage is being delivered to the relevant appliances. Overall this demonstration illustrates that our proposed HVAC control system is sufficiently working right with all the expected functionality and has met all requirements.

# 

# 5 Testing

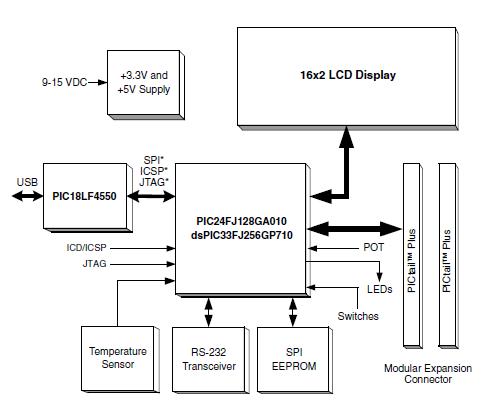
For the testing phase, we planned around having a full two weeks to test our unit on PCB and we hit the planned milestone ahead of time. Besides testing our schematics in both a simulator and on breadboards, we also wanted to have our PCB received and mounted for testing no later than 2 weeks before our presentation date. We were nearly full two weeks ahead of the planned milestone receiving our PCB almost four weeks before our planned presentation and had the components mounted the very same day. It did turn out that there were a small number of minor errors in our PCB however they were easily fixable and we have since updated our schematics and PCB design accordingly. On the software side we wanted to ensure every possible scenario was taken into account to ensure proper all of the important features such as power savings logic, heating and cooling, and scheduling were fully functional without bugs in the final working product.

## 5.1 Explorer 16 Development Board

Since most of the components of our entire system are developed by Microchip® Company, one obvious aspect that was realized was that we needed a development kit for testing and experimentation of the proposed design. The development kit needed to provide all of the sub systems and components functions for accurate testing and development. The Explorer 16 Development Board was selected and was more than adequate for this project’s needs. Further the output from the boards is displayed via the illumination of lights to make sure that the output signals are going through.

The Explorer 16 Development Board by Microchip® provides a low cost, modular development system for 16-bit microcontroller families. This board supports both the PIC24 microcontroller family which we used in the system for our Main Control Unit and Remote Sensor unit. The development board was the main testing device for our system.

This Microchip® Explorer development board offers many great features and capabilities that helped us with testing, prototyping and development of our system modules. The development board operates with a direct 9V DC power input, which it then powers down to the +3.3 V and +5 V needed to run our devices being tested. An LCD display on the board allows the tester to see real time results when programming the different modules on the board. Push button switches onboard are used for inputs and resets by the designated. There are also LED lights for various testing uses as well, which came in handy when running most of our debugging and testing. The high level block diagram in Figure 51 below illustrates the main connections between the integral parts and components.



**Figure 51 High Level Explorer 16**

**Development Board Block Diagram**

**(Reprinted with permission from Microchip**®**)**

The Explorer 16 Development board's primary functions are debugging and testing code for desired functionality of the system. A 6-wire In-Circuit Debugger (ICD) connector is included on the board to allow connection to be established with the MPLAB ICD 3 software included with the development kit. The MPLAB ICD 3 is a programming and debugging module made for beginning users to provide real time debugging, high speed programming and a ruggedized probe interface, perfect for protection against over voltage and current situations.

Two preprogrammed high-performance 16-bit digital signal controllers included with the Explorer 16 Development Board are a dsPIC33FJ256GP710 and a PIC24FJ128GA010. These included controllers were great additions to the kit since our main microcontroller unit is under the PIC24FJ128GA010 family. The Plug-in Modules (PIM) option on the board is an essential feature that allows for expansions to the main functionality provided. The secondary microcontroller is not included with the development kit, so having Microchip's® PIM options for the other PIC families was essential for our project. For the secondary microcontroller, we need to use the 100-pin Plug-in Module (PIM) for our PIC24F04KA201 for testing purposes. Microchip® has developed a PIM with a 28-pin PIC24F16KA102 sample device which can be used with the Explorer 16 Development board. This sample PIM is in the same related family as the PIC24FJ128GA010 which is the module we are going to use for our design.

### 5.1.1 Communicating with ARM

In our system testing for feasibility of communication between the main controller unit and the high level controller unit used the Explorer 16 Development Board. The connection between the two units is done through serial communication. The Explorer 16 board and the beagle board which contains an ARM microprocessor are connected using a serial cable to both devices' RS-232 transceivers ports that allow for the passing of data of which are used to send user inputted entries from the LCD screen display connected to the Pandaboard to the Explorer 16 board to validate the accuracy of data transfer before final implementation on the actual hardware. Furthermore implementing XBee shouldn’t be much more difficult in future version as it too uses RS-232 (via USB) however for more reliable control we opted for a wired solution. Future design considerations would include fail safes should wireless be implemented that allows for the system to still operate properly despite the loss of connectivity.

## 5.2 Safety Considerations

One issue that had to be acknowledged and prevented while designing and implementing our system is electrostatic discharge (ESD). ESD happens when electric current flows between two objects at different electrical potentials. These unwanted currents of ESD are the main culprit to electronic equipment damage. To prevent ESD from affecting all of our electrical parts and components, we implemented some plans to protect ourselves and the project from this nuisance. Grounding all conductive materials, objects and developers directly interfacing with the electronic equipment during the prototyping, developing and testing phases of the project is the best way to prevent ESD. Anti-static wrist straps are the most cost effective method to prevent ESD. We ordered two plus had an additional two from the group members owning anti-static wrist straps themselves, which allowed for all of us to work on circuits at the same time while eliminating potential damage to the circuits.

## 5.3 Equipment

To observe and test the behavior of our HVAC control system and its subsystems we are going to use the available devices that are readily available to us whether in the Senior Design Lab or other available labs. If some of the instruments are not there for us, we can always buy them from an independent vendor.

* **Oscilloscope** – An Oscilloscopeallows signal voltages to be viewed on a x-y coordinate system. Even though the oscilloscope shows voltages on the y-axis any other quantity can be changed to represent values such as current. Oscilloscopes are useful because it can display the shape of a particular wave. Each waveform can measure and present various characteristics such as amplitude, period, and frequency. Oscilloscope can also show are the pulse width and rise time of two distinct occurrences.
* **Function Generator** – A function generator is an electronic test tool used to make electrical signals or waveforms. Function generators create waveforms such as a sine wave, square wave, triangle wave, and saw tooth wave.
* **High Current Ammeter** – An ammeter is necessary for our HVAC control system because it was able to measure electrical current in a circuit. The device uses a moving coil with a needle which reads the amount of current being generated by the circuit.
* **Breadboard** – A breadboard is used as a construction setup or base for where we place integrated circuit chips in order to connect wires to different I/O ports. It was used to test the integrated chips so that they are working according to our provided data sheet specifications. A solder less breadboard was used as a prototype to test and observe the behavior of components such as LEDs, diodes, resistors, capacitors, and inductors. We used one large breadboard and two medium sized bread boards for initial component testing before mounting the parts on their proper locations on the PCB.
* **Multimeter** – A multimeter is an essential tool to use for our project because it can measure voltages, currents, and resistances. This electronic tool can be measured either with analog or digital circuits. Multimeters are also useful in figuring out the electrical problems in wiring systems and power supplies.
* **Computer Simulation** – Computer simulation is used as a preliminary basis for schematic layout design and printed circuit boards. It was useful as designers where every components should be placed and how much current and voltage is necessary for these particular parts.
* **Soldering Gun** – A soldering gun is a tool that is used to join loose wire connections in a circuit or network. Along with a soldering gun, solder thread is needed to tap out or heat solder filament.
* **Mouse and Keyboard** – For the initial testing of the ARM microprocessor we needed a keyboard and mouse. The touch screen does not provide a touchscreen keyboard in Linux terminal as there is not GUI operating system functionally operating. Furthermore calibrating the touchscreen was problematic without a keyboard and mouse also connected. To configure the device and build the Linux distribution properly we needed a mouse and keyboard.
* **Separate Linux Development Environment** – For the setup and configuration of the ARM microprocessor per the setup instructions we need a computer running Linux (we chose Ubuntu) that allows us to be able to communicate via RS-232 to the PIC microcontroller as well as format a flash memory card in a Linux compatible file format. The computer also then serves the dual purpose of being able to put the operating system on the flash memory card as it can read and write that file format.

## 5.4 Environment

The testing environment for the bulk of the development and testing time was spent inside the Senior Design II lab located in the Engineering Building 1 on the University of Central Florida (UCF) main campus. If other components that we purchase require more workspace to develop then our group will utilize selected off campus locations in order to construct our HVAC control system. Traveling to different sites to test out different systems supplied by the project sponsors wasn’t needed except in a few cases where we opted to do so. When placing our HVAC control system's remote sensor unit outside in an open environment exposed to the weather elements, it must have a cover to protect its electrical components in case of bad weather causing heavy amounts of precipitation or sever conditions. All facilities chosen for us to conduct our development or testing had a high standard of safety and procedures in case of electrical fire due to a wire being shorted or overloaded with too much current as well as good grounding which served the dual purpose of keeping group members alive and preventing equipment from being destroyed should there be a short. A good ground also prevents damage by providing a proper ground for the components itself as well as members themselves especially while using anti-static wrist guards to prevent electrostatic discharge on sensitive equipment. Furthermore many if not all of our components are Restriction of Hazardous Substances Directive (RoHS) compliant which is good should any fire happen we and others in our immediate area will not be exposed to as many toxic materials via pollutions of the air in the testing environment. Also upon disposing of destroyed parts the components will leech fewer chemicals into the environment. In this regard our testing environment is really everyone’s environment and as such we take our conduct there and its implications seriously.

## 5.5 Relays Testing

For testing the Pandaboard we will follow essentially the directions in the manual. We needed either a USB OTG Y-type style cable or a 5v 4A power supply – we opted for the 5V 4A power supply - an LCD monitor with DVI or HDMI input, a mouse, a keyboard, a Secure Digital flash memory card at least 4GB in size and a personal computer with both a Linux distribution installed - preferably Ubuntu - and an RS-232 port. We had format the flash memory card on the Linux personal computer with the proper Linux file formatting, in this case ext3. We will then place the test software included on the Pandaboard’s website, which tests the basic functionality of the device, on the flash memory card. We then took the flash memory card and insert it into the Pandaboard. We then connect all other external devices needed, LCD screen, mouse, and keyboard except the power. After double checking the connections we will then power on the device by plugging it in to the USB OTG port. Upon booting the program ran a series of diagnostic tests that will make sure all features of the board are working.

## 5.6 Relays Testing

There are multiple relays coming out of the main control board, and it was necessary to be sure that the units are working more than just checking for the familiar click of a relay switching. All these relays could be hooked up to an oscilloscope or multi-meter for testing 24VAC, but for demonstration purposes a different approach was used. It is impractical and too costly to hook up two air conditioners and a heater, so those three systems will be represented by lights powered by the 24VAC output. The fan output will be connected to a smaller fan that will run on 24VAC for effect. An electronic damper will be purchased and connected to one zone to show that the system can more than power that device. The mood scent will be implemented for the scope of this project as engineering a typical department store scent disperser to be connected and functional on 24VAC.

## 5.7 Final Specifications and Requirements

We completed and fully tested the main specifications of the unit as required by the sponsor. Additional secondary features were over 90% completed however the remaining 10% is easily implementable in future iterations, such as the ability to change the operation of the HVAC system remotely via the internet. Due to time constraints we were only able to implement the ability to read the current state of the HVAC system such as heating or cooling and the inside and outside temperature and humidity, as well as the inside CO2 levels. Our sponsor was willing to also provide additional funding beyond our initial $1500 budget and we came within the additional funding limit as well. Below in Table 16 is a complete part list with total prices.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Description** | **Part Number** | **Manufacturer** | **$ per unit** | **Quantity** | **Total $** |
| Inductor Power 22 uH 11.0A SMD | 732-2177-1-ND | Digi-Key | 8 | 2 | 16.00 |
| Diode Ult Dual 300V 10A | UH20FCT-E3/4WGI-ND | Digi-Key | 1.62 | 2 | 3.24 |
| CAP CER 10 uF 10V X5R 0603 | 587-2562-1-ND | Digi-Key | .536 | 10 | 5.36 |
| Relay PWR SPDT 8A 3VDC | Z2758-ND | Digi-Key | 3.53 | 15 | 52.95 |
| IC Diff Bus Transceiver 8-SOIC | 296-9693-5-ND | Digi-Key | .82 | 6 | 4.92 |
| Trans GP SS NPN 40V TO92 | P2N2222AGOS-ND | Digi-Key | .36 | 12 | 4.32 |
| Res 649 Ohm 1/10W 1% 0603 SMD | P649HCT-ND | Digi-Key | .04 | 20 | .80 |
| Res 470 Ohm 1/10W 5% 0603 SMD | P470GCT-ND | Digi-Key | .02 | 20 | .40 |
| Res 10K ohm 1W 5% 2512 SMD | PT10KXCT-ND | Digi-Key | .8 | 20 | 16.00 |
| Cap Alum 33uF 250V 20% Radial | P13534-ND | Digi-Key | .99 | 4 | 3.96 |
| Inductor Power 33uH 8.5A SMD | 732-2178-1-ND | Digi-Key | 8.00 | 5 | 40 |
| Diode Schottky 60V 3A C-16 | MBR360-ND | Digi-Key | .62 | 6 | 3.72 |
| Cap Alum 22uF 250V 20% Radial | P13532-ND | Digi-Key | .71 | 12 | 8.52 |
| IC Diff Bus TXRX 8-SOIC | 296-14636-1-ND | Digi-Key | .82 | 4 | 3.28 |
| IC Diff Bus TXRX 8-DIP | 296-1415-5-ND | Digi-Key | .82 | 4 | 3.28 |
| Cap Alum 120uF 35V | P11234-ND | Digi-Key | .48 | 16 | 7.68 |
| **Description** | **Part Number** | **Manufacturer** | **$ per unit** | **Quantity** | **Total $** |
| Res 6.2K Ohm 1/5W 0603 SMD | RHM6.2KDCT-ND | Digi-Key | .083 | 10 | .83 |
| Res 12.4K Ohm 1/10W 0603 SMD | P12.4KHCT-ND | Digi-Key | .04 | 10 | .40 |
| Res 5.62K Ohm 1/8W 0603 SMD | RNCP0603FTD5K62CT-ND | Digi-Key | .038 | 10 | .38 |
| IC Reg Simple Switcher | LM2679S-3.3-ND | Digi-Key | 7.8 | 3 | 23.40 |
| IC Reg Simple Switcher | LM2679S-12-ND | Digi-Key | 7.8 | 2 | 15.60 |
| Cap Cer .1uF 25V 0603 | 490-1575-1-ND | Digi-Key | .026 | 20 | .52 |
| Temp/Hum Sensor | SHT21 | Liquidware | 39.93 | 3 | 143.51 |
| VPack5.0V\_AA |  | Bodhilabs | 10.95 | 1 | 11.95 |
| 3A Diodes | 1N5400 | Futurlec | .10 | 20 | 24 |
| RJ11 6-Pin Connector | PRT-00132 | Sparkfun | 1.25 | 4 | 5 |
| 9-Pin Female Serial Connector | PRT-00429 | Sparkfun | 1.5 | 2 | 3 |
| RS232 Converter | COM-00589 | Sparkfun | 1.95 | 3 | 5.85 |
| RJ45 8 pin Connector | PRT-00643 | Sparkfun | 1.5 | 4 | 6 |
| Screw Terminals 3.5 mm pitch | PRT-08084 | Sparkfun | 1.13 | 10 | 11.30 |
| Jumper Wire | PRT-08599 | Sparkfun | 1.50 | 2 | 3 |
| JST Vertical Connector | PRT-08613 | Sparkfun | 0.95 | 4 | 3.80 |
| Jumper 2 Pin | PRT-09044 | Sparkfun | 0.32 | 10 | 3.20 |
| Basic LED | COM-09592 | Sparkfun | 0.32 | 10 | 3.20 |
| Secondary PCB |  | 4PCB | 61.32 | 1 | 61.32 |
| Main PCB |  | 4PCB | 33 | 2 | 66 |
| Electrical Test |  | 4PCB | 99 | 1 | 99 |
| Shipping Charges |  |  |  |  | 43.65 |
| ADAPTER DESKTOP CLASS | 993-1019-ND | Digikey | 27.67 | 1 | 27.67 |
| LINE CORD 3 COND US | 993-1039-ND | Digikey | 3.18 | 1 | 3.18 |
| Shipping |  |  |  |  | 18.91 |
| **Description** | **Part Number** | **Manufacturer** | **$ per unit** | **Quantity** | **Total $** |
| RES 10K OHM SMD | P10KGCT-ND | Digikey | 0.02 | 10 | 0.2 |
| DIODE SCHOTTKY 50V 3.3A | 31DQ05-ND | Digikey | .805 | 10 | 8.05 |
| CAP ALUM 180UF | P10245-ND | Digikey | .49 | 18 | 8.82 |
| CAP ALUM 15UF 50V | P10317-ND | Digikey | .33 | 24 | 7.92 |
| INDUCT PWR 25UH 3.0A | 553-1424-ND | Digikey | 1.89 | 8 | 15.12 |
| CAP CERM 1000PF | 478-1215-1-ND | Digikey | .02 | 10 | 0.2 |
| CAP CERM .01UF | 478-1227-1-ND | Digikey | .018 | 10 | 0.18 |
| CAP CER .47UF 50V | 445-5952-1-ND | Digikey | .275 | 10 | 2.75 |
| CAP ALUM 120UF 35V | P11234-ND | Digikey | .48 | 16 | 7.68 |
| RES 6.2K OHM 1/5W SMD | RHM6.2KDCT-ND | Digikey | .083 | 10 | .83 |
| RES 12.4K OHM 1/10W SMD | P12.4KHCT-ND | Digikey | .04 | 10 | .40 |
| RES 5.62K OHM 1/8W SMD | RNCP0603FTD5K62CT-ND | Digikey | .038 | 10 | .38 |
| IC REG SIMPLE SWITCHER | LM2679S-3.3-ND | Digikey | 7.8 | 3 | 23.40 |
| IC REG SIMPLE SWITCHER | LM2679S-12-ND | Digikey | 7.8 | 2 | 15.60 |
| CAP CER .1UF | 490-1575-1-ND | Digikey | .026 | 20 | .52 |
| Shipping |  |  |  |  | 23.17 |
| dsPIC33 GP 100P to 100P TQFP Plug-In Module | MA330011 | Microchip | 18.75 | 1 | 26.75 |
| Xenarc |  | Xenarc Direct | 349 | 1 | 349 |
| PandaBoard etc. |  |  |  |  | 275 |
| XBee module |  | Sparkfun | 22 | 3 | 66 |
| XBee to USB |  | Gravitech | 20 | 1 | 20 |
| Sandisk MicroMate  Memory Stick |  | Amazon | 4.25 | 1 | 4.25 |
| **Description** | **Part Number** | **Manufacturer** | **$ per unit** | **Quantity** | **Total $** |
| Anti-Static Wrist Band |  | Belkin | 7 | 2 | 14 |
| WINTEC FileMate 16GB |  | Wintec | 25 | 1 | 25 |
| SOLDERLESS BREADBOARD |  | MPJA | 12 | 2 | 24 |
| SOLDERLESS BREADBOARD  & JUMPERS |  | MPJA | 19 | 1 | 19 |
| Sockets |  | Home Depot | 4 | 3.99 | 15.96 |
| GEClear Tube |  | Home Depot | 3.89 | 1 | 3.89 |
| 7WNHTLIT |  | Home Depot | 3.37 | 1 | 3.37 |
| Plastic Cutter |  | Home Depot | 3.99 | 1 | 3.99 |
| C7 LED |  | Home Depot | 4.97 | 1 | 4.97 |
| SD BUT |  | Home Depot | 5.99 | 1 | 5.99 |
| .093-11X14AC |  | Home Depot | 3.98 | 1 | 3.98 |
| PlastBaggos |  | Home Depot | 1.96 | 2 | 3.92 |
| ADJSCKTCB2 |  | Home Depot | 3.99 | 3 | 11.97 |
| C7 LED |  | Home Depot | 4.97 | 5 | 24.85 |
| XFMR |  | Radioshack | 12.99 | 1 | 12.99 |
| Lamps |  | Radioshack | 2.19 | 1 | 2.19 |
| PCB Term 2P 5MM |  | Radioshack | 2.39 | 1 | 2.39 |
| Transformer |  | Skycraft | 11 |  | 22 |
| Perfboard |  | Skycraft | 2.95 | 1 | 2.95 |
| Rectifier |  | Skycraft | 4.95 | 1 | 4.95 |
| TOTAL |  |  |  |  | 1827.68 |

**Table 15 Parts List**

### 5.7.1 Final Specifications

Based on meetings with our sponsors, we were able to produce a set of specifications to model our Efficient HVAC Control and Feedback System with. We were able to complete nearly all of the final specifications by our extensive research and design work and we have come out very similar to our initial specifications. We realized the for any specification whether it be for a sensor or control unit that there was a vast amount of choices from many vendors and suppliers which made any requirement of the system possible but with detailed research of the alternatives. We could have spent many more months researching and deciding which parts were the best to choose. There were many parameters to consider which parts. They varied from ease of use, availability, and cost. These specifications are considered final, but there is a chance the production version might have different parts and changes occur due to any number of reasons, and our varied research in the many subjects above will prove to be valuable as a change of parts or direction won’t immediately change the impact of the design of the. The overall cost of the system built with these specifications will not be indicative of the cost of building a final production system as much of what we are using is for development. Often development boards accrue a much higher cost. The final specifications we have come up with for the Efficient HVAC Control and Feedback system are listed below in both Table 16.

|  |  |
| --- | --- |
| **Total Cost of the system materials (including development kit)** | <$1500 |
| **Intersystem Wireless data transmission distance** | 100 feet or more |
| **Wireless communication standard for Remote control of the system via mobile device** | 802.11b wireless communication |
| **Control voltage to entire HVAC components** | 24V AC |
| **Realistic temperature | Relative humidity for exterior environment measurements** | (0°F - 110°F) | (0% - 100%) |
| **Accuracy of the Measurements of interior temperature** | +/-0.5°C |
| **Accuracy of the Measurements of exterior temperature** | +/-0.5°C |
| **Accuracy of the Measurements of interior relative humidity** | +/- 5% |
| **Accuracy of the Measurements of exterior relative humidity** | +/- 5% |
| **LCD screen size** | 7 inch |
| **Operating System Environment** | Ubuntu Linux |

**Table 16 System Control Specs**

The two microcontrollers handle all the real time information and interaction with the electrical components. The main microcontroller, while much less powerful than the ARM unit, provides the instruction set for our system. The secondary controller is used to transmit the temperature and humidity data wirelessly. Their specifications are given below in Table 17.

|  |  |
| --- | --- |
| **Main Microcontroller** | **Secondary Microcontroller** |
| Supply Voltage: 3 - 3.6V | Supply Voltage: 1.8 - 3.6V |
| Up to 40 MIPS | 4 Kbytes program memory |
| Two 40 - bit accumulators | 10 - bit A/D converter |
| 32/16 and 16/16 divide operations | C Language compatible |
| 16 x 16 fractional/integer multiply operations |
| 256 Kbytes Flash memory | 4 Kbytes program memory |
| 30 Kbytes RAM | 512 bytes RAM |
| 85 programmable I/O pins | 18 programmable I/O pins |
| 2 SPI ports | 1 SPI port |
| 2 I2C ports | 1 I2C port |
| 2 UART ports | 1 UART port |

**Table 17 Final Microcontroller Specs**

The main microcontroller that is used in the design requires a supply voltage of 3 – 3.6 V in order to be powered on. It has 85 programmable I/O pins that will be used to support control relays as outputs. It has 2 SPI ports, 2 I2C ports, and 2 UART ports that will be used for interfacing with other components in the main control unit. The 256k bytes of on-board flash memory will be used to store the programming for the decision making logic and the 512 bytes RAM will run the application to send data to high level control unit. All programming for the main microcontroller was done using the C language.

The secondary microcontroller that is used in the design requires a supply voltage of 1.8 – 3.6V in order to be powered on. It has 1 SPI port and 1 I2C port that will be used to communicate with a sensor and the XBee chip onboard the remote sensor unit. The programming will be stored in the 4K bytes program memory and the 512 bytes RAM will run the program. All programming for the secondary microcontroller was done using the C programming language.

### 5.7.2 Final Requirements

The sponsor for our project, AC3 Development Group, LLC provided the basic requirements for our project and allowed for added flexibility with the removal or addition of specifications based on time constraints and feasibility. At a high level, the basic requirements of the system were very simple and obtainable based on the first version of the HVAC Feedback and Control System however they were poorly done so in the first version. The system is required to sense the temperature and relative humidity from inside and outside a structure (residential or commercial), read the settings determined by the user (inputted via LCD touch screen interface), and make anintelligent decision to satisfy the preset temperature and relative humidity levels in the most effective and energy efficient manner. Some of these features were implemented in the first version of the system and were implemented in this iteration. For this iteration of the system, it was required to have a scheduler to store user submitted entries of desired temperature and humidity settings at specified interval throughout a 7 Days week span. An authorized technician is also able to log into the system and adjust the available HVAC components and the features of the application along with perform system software update.

The system is intended to either be retrofitted to an existing HVAC system, or to be installed in a new building but only require essentially the normal wiring that would be installed for a normal HVAC system. In order to avoid running additional wire not normally associated with an HVAC system inside either an existing structure or a newly constructed building the temperature and relative humidity measurements taken outside is transmitted to the Main Control Unit wirelessly. In order to achieve wireless data transmission, the sensor takes its readings; passes the data to a microcontroller which, in turn, then passes the data to a wireless chip that communicates with another wireless chip on the Main Control Unit which then sends data to and receives data from the High Level Control Unit.

The wiring for the air conditioning units to be controlled is already installed in the building, and therefore controlling the power relays to turn the air conditioning units on and off can be done via wire. AC units typically run on 240 Volts AC. To get this voltage to the units, a power relay must be switched to essentially complete a circuit connecting the units to the main 240V power supply from the building. Out control system sends a signal to switch appropriate relays to send 24 V to the 240 V power relays which turn on the AC units. To control each unit, a signal must be used for each of the following functions of the unit. The HVAC components of an air conditioning system are, 1st Stage Heating, 2nd Stage Heating, Blower / Fan, Compressor, Reversing Valve, and 24V Common. The American Society of Heating, Refrigerating, and Air-Conditioning Engineers set standard naming and color coding standards for these functions that we plan to follow in both our system design and the planned demonstration. Table 18 shows each HVAC Component along with its ASHRAE (American Society of Heating, Refrigerating and Air-Conditioning Engineers) Naming and Color Coding Standard.

|  |  |
| --- | --- |
| **HVAC Component** | **ASHRAE Naming & Color** |
| 1st Stage Heating | W1 |
| 2nd Stage Heating | W2 |
| Blower/Fan | G |
| Compressor | Y1 |
| Reversing Valve | O |
| 24V Common | R |

**Table 18 HVAC Components and the Corresponding ASHRAE Naming and Color Standards**

The subsystem that controls new air flow into the building is required to control four dampers in the ducts of the HVAC system, and is controlled wirelessly. These four dampers will control airflow through the ducts depending on user controlled settings via the touch screen. If a dehumidification unit is not integrated into the existing HVAC system at the time the Efficient HVAC Control and Feedback System is installed, it must also be controlled wirelessly.

# 6 Administrative

We were very wise to make sure we made tight milestones and ensure we met those milestones so we could be ahead should we ever run into unforeseen circumstances that might take more time than we anticipated. We were also wise to keep good records of the parts we ordered. To save money, we cataloged our initial parts that we started off with so we didn’t double ordered parts we already had. We also kept weekly logs of each individual group member’s progress.

## 6.1 Milestones discussion

Our intent was to start off strong and finish stronger. The intent was to chose a project and stick with it. We could have continued searching for weeks to find the perfect project that compliments all of our skills, and never find it. We decided a strong decision making campaign was more important and immediately chose this project as the first group to make a decision. Upon making our decision we portioned out our intended milestones for senior design one. The first week was dedicated, through three separate meetings, on how we would portion out the time and succeed at this project. Our first week was a success and we met all the milestones up to that point. We entered research on schedule and intended to have all research done by mid-February or at the latest spring break. This goal proved to be slightly over ambitious as we continued research throughout the entire semester, even while designing our device. Our core research was done, though, by spring break as we knew our total goals and how to implement them by then. Design and research overlapped but was met on time. Design proved to be the more difficult of the milestone goals, as we found our quick and determined decision making challenged with doubt in some products ability to be supplied or adequate support being added. We spent a good portion of time working on the connectivity of the microcontrollers. Due to the fact that we are all not very experienced with microcontrollers, this contributed to the uncertainties that we faced.

Upon consulting with very knowledgeable individuals, we finally made the decision on our microcontroller and its connectivity. Further, finding the right sized LCD touch screen that was affordable and compatible proved to be somewhat exhausting but decisions had to be made. The sponsor made it clear to us that we could go ahead and purchase a relatively more expensive LCD but due to the fact we want to achieve the goal of low cost, it took us lots of deliberation to decide officially which LCD screen we are going to use. One of the requirement that we most definitely have is that the screen should have a good resolution which is the same or better than version 1.0. We believe that it was important to make our GUI appealing to the eye of the customer and sponsor as this is usually a factor in someone purchasing a product. The power system was slightly easier, though powering devices with 24VAC is not a commonality in most cases and those parts had to be found with the proper tolerances to move forward. We needed to be sure that our power supply design is working according to our need which would make our project cosmetically and structurally sound. We have put this much focus into this section of the project simply because of the mistakes that the previous team went through with their power supply. This did get fully finished though as we have full confidence in the parts that we have chosen above in the final product as they are all working. Prototyping this design was easier than intended on paper. The extra time dedicated to researching connectivity standards and ports such as UART, I2C, SPI, etc. all paid its dividends here. We feel very confident heading into senior design two that we set ourselves up for success with little need of backtracking, sending us steps back into a research and design infinite loop. We hoped to get started on our embedded software immediately upon completion of the semester and luckily we met that deadline. The fact that we are connecting the ARM to the PIC via RS-232, we wanted to spend as much time in the lab at an early time getting our code to work to avoid future problems. By doing these tasks early it prevented us from having to panic at the end of next semester because our project doesn’t work. Our extensive testing of our sensors under various situations and with sudden disruption of power in the last few weeks was also very helpful. Our goal to debug intensively so that a vast majority of the bugs are out of the system by presentation day was successful.

## 6.2 Budget Discussions

We were very fortunate to have a $1500 budget. For our project that figure to work proved excellent. Given our fortune here in our extensive testing it appears from our research that our unit will cost slightly more than the original prototype unit. We are tried our best to remain within that range, but with given permission we went slightly above. The key in this project is just not getting it down at a low cost, but the four of us graduating from UCF by having a functional unit. The original PIC development kit has been acquired, saving us the cost of acquiring a new development board. Further the power supplys that were purchased will not be needed as the board will now be adequately designed to accept the input of the wall. The ARM board and LCD screen purchase appears so far to contain the same cost as the original unit approximately, but with much added flexibility.

**Bibliography**

* “Ajax (programming) - Wikipedia, the free encyclopedia”, n.d., http://en.wikipedia.org/wiki/AJAX.
* “CDC - Indoor Environmental Quality: Building Ventilation - NIOSH Workplace Safety and Health Topic”, n.d., http://www.cdc.gov/niosh/topics/indoorenv/BuildingVentilation.html.
* Glenn Corbett, “Fire Engineering’s Handbook for Firefighter I and II”, n.d., 326.
* “Flash point - Definition and More from the Free Merriam-Webster Dictionary”, n.d., http://www.merriam-webster.com/dictionary/flash+point?show=0&t=1302977722.
* “Flashover - Definition and More from the Free Merriam-Webster Dictionary”, n.d., http://www.merriam-webster.com/dictionary/flashover.
* “HTML5 - Wikipedia, the free encyclopedia”, n.d., http://en.wikipedia.org/wiki/HTML5.
* “JavaScript - Wikipedia, the free encyclopedia”, n.d., http://en.wikipedia.org/wiki/JavaScript#Use\_in\_web\_pages.
* “TCP/IP model - Wikipedia, the free encyclopedia”, n.d., http://en.wikipedia.org/wiki/TCP/IP\_model.
* “thttpd - Wikipedia, the free encyclopedia”, n.d., http://en.wikipedia.org/wiki/Thttpd.
* “View topic - AVR Studio 5 Released - Get Your BETA Here! :: AVR Freaks”, n.d., http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=103949&postdays=0&postorder=asc&start=120.
* “World Wide Web Consortium (W3C)”, n.d., http://www.w3.org/.
* “February 2009 Web Server Survey | Netcraft”, n.d., http://news.netcraft.com/archives/2009/02/18/february\_2009\_web\_server\_survey.html.
* “NHC RSS Feeds”, n.d., <http://www.nhc.noaa.gov/aboutrss.shtml>.
* Glenn Corbett, “Fire Engineering’s Handbook for Firefighter I and II”, n.d., 326.
* “iOS Developer Program License Agreement”, n.d., <http://developer.apple.com/programs/terms/ios/standard/ios_standard_agreement_20110215.pdf>.
* 706TSA - 7" TFT LCD Touchscreen Monitor w/ DVI & VGA & AV inputs. Xenarc Technologies Corp. < <https://www.xenarcdirect.com/product.php?productid=16194>>
* “The Effect of Short Cycling and Fan Delay on the Efficiency of a Modified Residential heat Pump”, n,d. <http://www.fire.nist.gov/bfrlpubs/build86/PDF/b86002.pdf>
* “Universal asynchronous receiver/transmitter - Wikipedia, the free encyclopedia”, n.d., http://en.wikipedia.org/wiki/Universal\_asynchronous\_receiver/transmitter.
* “I²C - Wikipedia, the free encyclopedia”, n.d., <http://en.wikipedia.org/wiki/I2c>.
* LS6410 S3C6410 ARM11 Android Development Kit (4.3" touchscreen). Linksprite Technologies, Inc. <http://www.linksprite.com/product/showproduct.php?id=25&lang=en>
* Mobile Web. Wikipedia The Free Encyclopedia. <http://en.wikipedia.org/wiki/Mobile\_Web>
* BeagleBoard-xM Product Details. beagleboard.org. <http://beagleboard.org/hardware-xM>
* Xenarc Technologies-Corporate Info. Xenarc Technologies, Corporation
* Corporate Overview. Microchip Technology, Inc. <http://www.microchip.com/stellent/idcplg?IdcService=SS\_GET\_PAGE&nodeId=72>
* Company Portrait. Sensirion. <http://www.sensirion.com/en/05\_company/00\_portrait.htm>

**Abbreviations and Acronyms**

AC – Alternating Current

ACK – Acknowledgement Frame

ADC – Analog to Digital Converter

ASHRAE – American Society of Heating, Refrigerating and Air-Conditioning Engineers

ARM – Advanced RISC Machine

BTU – British Thermal Unit

CDC – Center for Disease Control

CpE – Computer Engineering

CPU – Central Processing Unit

DC – Direct Current

EE – Electrical Engineering

EEPROM – Electrically-Erasable Programmable Read-Only Memory

GHz – Gigahertz

HVAC – Heating, Ventilation, and Air Conditioning

I/O – Input/Output

I2C (or I2C) – Inter-Integrated Circuit)

IC – Integrated Circuit / Integrated Chip

IDE – Integrated Development Environment

ISA – Industry Standard Architecture

ISM – Industrial, Scientific and Medical

LCD – Liquid Crystal Display

MAC – Media Access Control

Mbps – Megabits per second

MCU – Main Control Unit

MIPS – Millions of Instructions Per Second

NC – Normally Closed

NVRAM – Nonvolatile Random Access Memory

PAN – Personal Area Network

PC – Personal Computer

PCB – Printed Circuit Board

PPM – Parts Per Million

RAM – Random Accessing Memory

RF – Radio Frequency

RoHS - Restriction of Hazardous Substances Directive

ROM – Read Only Memory

SPI – Serial Peripheral Interface

SRAM – Static Random Access Memory

UART – Universal Asynchronous Receiver/Transmitter

UCF – University of Central Florida

USB – Universal Serial Bus

Wi-Fi – Wireless Local Area network

XLP – Extreme Low Power

**Appendix A: Datasheets**

BeagleBoard-xM Board User Guide by BeagleBoard

Overa Board User Guide by Gumstix

BeagleBoard-xM Schematic by BeagleBoard

PandaBoard Board User Guide by Panda Board

PandaBoard Schematic by Panda Board

Explorer 16 Development Board User‟s Guide by Microchip

dsPIC33FJXXXGPX06A/X08A/X10A Data Sheet by Microchip

PIC24F04KA201 Family Data Sheet by Microchip

SHT21 Datasheet by Sensirion

SRS-03VDC-SL Datasheet by Songle Relay

CO2 Engine K22 OC by Senseair

CO2 Engine K30 by Senseair

CO2 Engine K33 BLG/ ELG Senseair

MRF24WB0MA/MRF24WB0MB Datasheet by Microchip

MRF24J40 Datasheet by Microchip

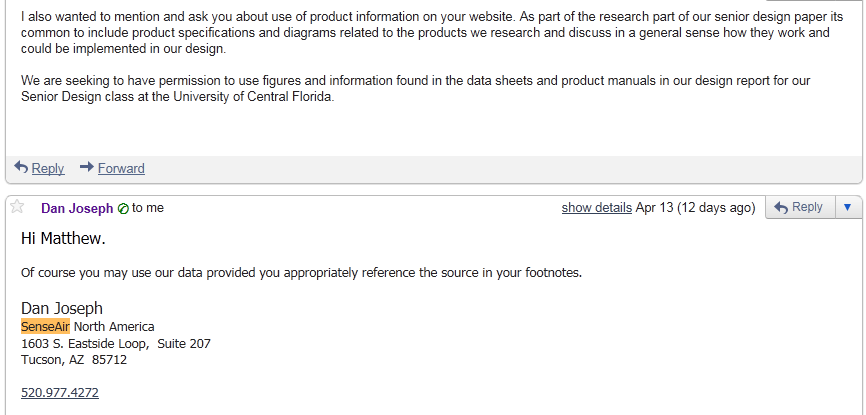
DM3730 BeagleBoard-xM Rev A3 System Reference Manual by Texas Instruments

**Appendix B: Copyright / Permissions**

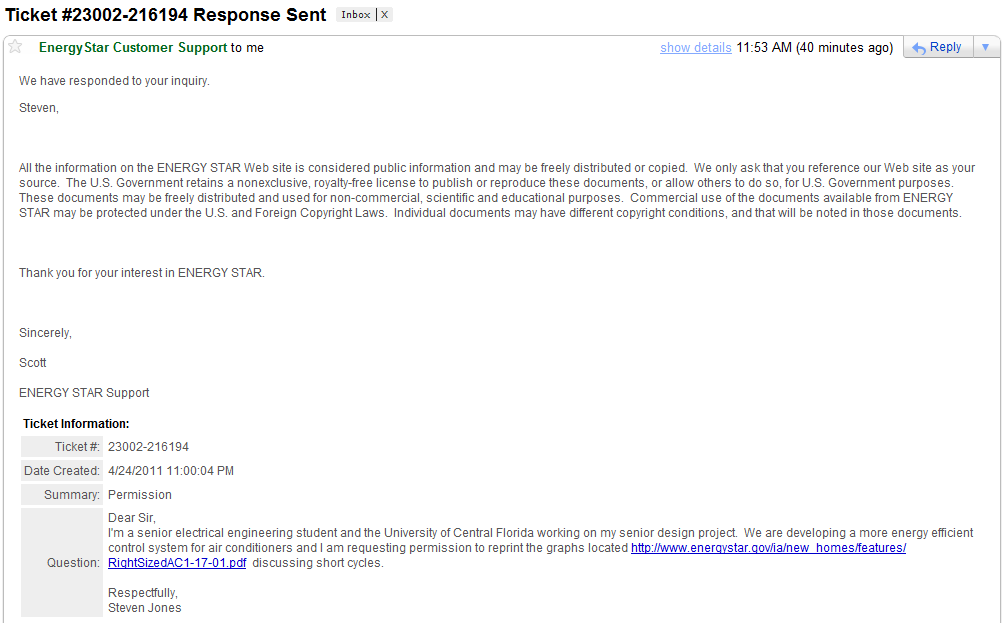
Information on the creative commons license:

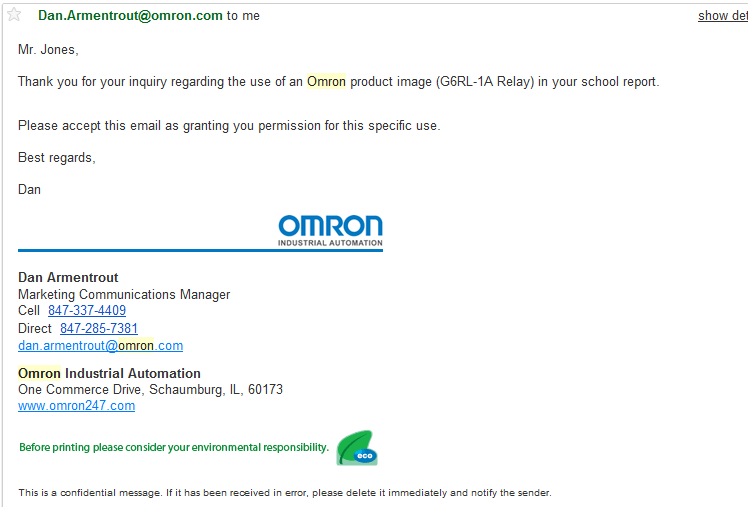
“All Creative Commons licenses have many important features in common. Every license helps creators — we call them licensors if they use our tools — retain copyright while allowing others to copy, distribute, and make some uses of their work — at least non-commercially.”

More information at: <http://creativecommons.org/>









PENDING:

