VOICE CONTROLLED HOME AUTOMATION
Technical Report – Voice Controlled Home Automation

Richard McAlpine

Professor: Robert Woudsma
Date: April 21 2009
Executive Summary

The primary purpose of this report is to display how one may implement a voice controlled home automation system into a given house. A demo system has been created to show the functionality of such a system. As technology is continuously becoming more advanced and available, an automated system that is controlled by voice is not a farfetched idea in today’s world. This demo system will show that by combining very simple and accessible components, anyone could have this system today.

The system uses two different software programs and various hardware components. The Interface is software designed using Visual Basic that displays the needed information (room temperature etc.), performs the PID algorithm for temperature control, and is what the user will interact with either directly or indirectly. The SayNow software provides the voice recognition component of the system. The computer receives and sends information to a small micro controller known as the Basic Stamp2. All the controlled components are wired into the Basic Stamp2 allowing control over lights, blinds, temperature, and alarms.

Wiring diagrams are provided to show how each component is installed and how this system is implemented in a sample apartment. The design could be easily modified to fit any home. Potential future upgrades are provided to discuss upgrades that were not properly tested but could be implemented into the system, including automatic magnetically locked doors, moving security cameras and additional interface capabilities.

The total cost of the System is $2570.79. The cost serves as a reference as this system is component based and is fully upgradeable.
# Table of Contents

1.0 Introduction .......................................................................................................................... 1

2.0 Components .......................................................................................................................... 3

2.1 The Home PC ....................................................................................................................... 4

   2.1.1 Description .................................................................................................................. 4

   2.1.2 Purpose in the system ............................................................................................... 5

   2.1.3 Installation .................................................................................................................. 5

2.2 The Interface ......................................................................................................................... 6

   2.2.1 Description ................................................................................................................ 6

   2.2.2 Purpose in the system ............................................................................................... 6

   2.2.3 Installation .................................................................................................................. 7

2.3 The SayNow Software .......................................................................................................... 7

   2.3.1 Description ................................................................................................................ 7

   2.3.2 Purpose in the system ............................................................................................... 8

   2.3.3 Installation .................................................................................................................. 8

2.4 The Voice Tracker Array Microphone .................................................................................. 8

   2.4.1 Description ................................................................................................................ 8

   2.4.2 Purpose in the system ............................................................................................... 9

   2.4.3 Installation .................................................................................................................. 9

2.5 SoundMax SoundBeam Array Microphone ......................................................................... 9

   2.5.1 Description ................................................................................................................ 9

   2.5.2 Purpose in the system ............................................................................................... 10

   2.5.3 Installation .................................................................................................................. 10

2.6 The Basic Stamp2 Microcontroller ..................................................................................... 10

   2.6.1 Description ................................................................................................................ 10

   2.6.2 Purpose in the system ............................................................................................... 11

   2.6.3 Installation .................................................................................................................. 11

2.7 The Stamp Interface Box ..................................................................................................... 12

   2.7.1 Description ................................................................................................................ 12

   2.7.2 Purpose in the system ............................................................................................... 13

   2.7.3 Installation .................................................................................................................. 13
2.8 The Sharp 8A Solid State Relay ................................................................. 14
  2.8.1 Description .......................................................................................... 14
  2.8.2 Purpose in the system .......................................................................... 14
  2.8.3 Installation ......................................................................................... 15
2.9 The PIR Sensor ...................................................................................... 16
  2.9.1 Description ........................................................................................ 16
  2.9.2 Purpose in the system ........................................................................ 16
  2.9.3 Installation ......................................................................................... 16
2.10 Sensiron Temperature/Humidity Sensor ................................................... 17
  2.10.1 Description ...................................................................................... 17
  2.10.2 Purpose in the system ...................................................................... 18
  2.10.3 Installation ...................................................................................... 18
2.11 The CRYDOM Proportional Power Controller ........................................... 19
  2.11.1 Description ...................................................................................... 19
  2.11.2 Purpose in the system ................................................................... 19
  2.11.3 Installation ..................................................................................... 19
2.12 The Parallax (Futaba) Continuous Rotation Servo .................................... 20
  2.12.1 Description ...................................................................................... 20
  2.12.2 Purpose in the system ................................................................... 20
  2.12.3 Installation ..................................................................................... 21
2.14 The Line Pulley Mechanism .................................................................. 22
  2.14.1 Description ...................................................................................... 22
  2.14.2 Purpose in the system ................................................................... 22
  2.14.3 Installation ..................................................................................... 22
2.15 The Turn Mechanism ........................................................................... 23
  2.15.1 Description ...................................................................................... 23
  2.15.2 Purpose in the system ................................................................... 23
  2.15.3 Installation ..................................................................................... 23
2.15 The LED Automatic Closet Light .............................................................. 24
  2.15.1 Description ...................................................................................... 24
  2.15.2 Purpose in the system ................................................................... 24
  2.15.3 Installation ..................................................................................... 24
3.0 The Demo System ........................................................................................................... 26
   3.1 The Interface .................................................................................................................. 26
      3.1.1 Interface display functions ....................................................................................... 27
      3.1.2 Interface Communications ....................................................................................... 30
      3.1.3 The PID algorithm .................................................................................................. 31
   3.2 Voice Commands ........................................................................................................... 33
   3.3 Communication protocols .............................................................................................. 34
      3.3.1 The Stamp 1 Protocol ............................................................................................. 34
      3.3.2 The Stamp 2 Protocol ............................................................................................. 35
      3.3.3 The Stamp 3 Protocol ............................................................................................. 36
      3.3.4 The Stamp 4 Protocol ............................................................................................. 37
   3.4 Light control .................................................................................................................. 38
   3.5 Blind control .................................................................................................................. 39
   3.6 Temperature control ...................................................................................................... 39
      3.6.1 Acquiring and interpreting the current temperature ................................................. 39
      3.6.2 Controlling the heating element .............................................................................. 39
   3.7 The demo system wiring ............................................................................................... 41
4.0 The Voice Controlled Home Automation System .............................................................. 42
   4.1 The Apartment .............................................................................................................. 42
   4.2 The Control Wiring ........................................................................................................ 43
      4.2.1 Motion Sensors ........................................................................................................ 44
      4.2.2 Alarms .................................................................................................................... 45
      4.2.3 Blind control .......................................................................................................... 46
      4.2.4 Temperature Measurement ..................................................................................... 46
      4.2.5 Temperature Control ............................................................................................... 47
      4.2.6 Lighting .................................................................................................................. 47
   4.3 The AC wiring ................................................................................................................. 48
   4.4 Microphones .................................................................................................................. 49
   4.5 The Bigger Picture .......................................................................................................... 50
5.0 Potential Future Upgrades ................................................................................................. 51
   5.1 Automatic magnetically locked doors ............................................................................ 51
   5.2 Moving Security Cameras .............................................................................................. 51
5.3 Additional Interface Capabilities .......................................................... 52
  5.3.1 Sending alerts to the user ................................................................ 52
  5.3.2 Redesigning the voice recognition feature ................................. 52
  5.3.3 Remote Control ........................................................................... 53
6.0 Conclusion ............................................................................................ 54
7.0 References ............................................................................................ 55
8.0 Glossary .................................................................................................. 56
9.0 Appendices ............................................................................................ 59
  Appendix A .............................................................................................. 59
    Appendix A.1 Voice commands and functions .................................. 59
  Appendix B .............................................................................................. 62
    Appendix B.1 Interface code .............................................................. 62
    Appendix B.2 Stamp 1 Code ............................................................... 70
    Appendix B.3 Stamp 2 Code ............................................................... 76
    Appendix B.4 Stamp 3 Code ............................................................... 79
    Appendix B.5 Stamp 4 Code ............................................................... 81
  Appendix C .............................................................................................. 81
    Appendix C.1 Wiring transparencies .................................................. 81
List of Figures

Figure 1 The home PC ................................................................. 4
Figure 2 The interface .................................................................. 6
Figure 3 The SayNow interface ...................................................... 7
Figure 4 Voice tracker array microphone (1) ................................. 8
Figure 5 SoundMax soundbeam array microphone ....................... 9
Figure 6 Basic Stamp 2 (2) .......................................................... 10
Figure 7 Stamp interface box ....................................................... 12
Figure 8 Voltage-conditioning circuit ........................................... 13
Figure 9 Stamp I/O to stamp interface box configuration ............... 14
Figure 10 Sharp 8A solid-state relay (2) ......................................... 14
Figure 11 Sharp 8A solid-state relay component wiring ................ 15
Figure 12 PIR Sensor (2) ............................................................. 16
Figure 13 PIR sensor component wiring ....................................... 17
Figure 14 Sensiron temperature/humidity sensor (2) ..................... 17
Figure 15 Sensiron temperature/humidity sensor component wiring. 18
Figure 16 CRYDOM proportional power controller (3) .................. 19
Figure 17 Proportional Power Controller Wiring ........................... 20
Figure 18 Parallax (Futaba) Continuous Rotation Servo (2) .......... 20
Figure 19 Servo component wiring .............................................. 21
Figure 20 Line pulley mechanism ................................................ 22
Figure 21 Turn mechanism ........................................................ 23
Figure 22 LED automatic closet light (4) ....................................... 24
Figure 23 Connection to LED ....................................................... 24
Figure 24 The interface with identification numbers ....................... 27
Figure 25 PID algorithm Visual Basic code ................................... 32
Figure 26 Interface to stamp 1 ...................................................... 34
Figure 27 Stamp 1 receive from interface ..................................... 35
Figure 28 Stamp 2 to interface ..................................................... 35
Figure 29 Interface receive from stamp 2 ...................................... 36
Figure 30 Stamp 3 to interface ..................................................... 36
Figure 31 Interface receive from stamp 3 ...................................... 37
Figure 32 Interface to stamp 4 ...................................................... 37
Figure 33 Stamp 4 receive from interface ..................................... 38
Figure 34 Demo System Wiring .................................................... 41
Figure 35 The apartment with all components ............................... 42
Figure 36 The apartment ............................................................. 43
Figure 37 The apartment with control wiring and component placement .................................................. 44
Figure 38 The apartment with AC wiring, light control wiring, and furniture .................................................. 48
Figure 39 The apartment with microphone placement .................... 50
Figure 40 Cost analysis table ....................................................... 54
Figure 41 Voice commands and functions ................................................................. 61
Figure 42 Microphone placement ............................................................................. 82
Figure 43 Furniture placement ................................................................................ 83
Figure 44 Light control wire placement .................................................................. 84
Figure 45 AC wiring placement ................................................................................ 85
Figure 46 Control wiring placement ......................................................................... 86
Figure 47 Component placement ............................................................................. 87
Figure 48 Apartment ................................................................................................. 88
1.0 Introduction

Home automation is not a new concept and many variations of it are employed today. It can enrich people’s lives by providing safety, security, and efficiency. Home automation can be as simple as turning on a light with a remote or as complex as automatically monitoring and controlling the heating and air quality of a home. This report will document the construction of an original home automation system that will be implemented in the author’s apartment. This system involves components that come from many different sources with the attempt of providing quality control that is both cost effective and user friendly. The system is designed to be fully upgradable, allowing the addition or removal of components as desired. The system is designed to interface with a Windows XP home PC. By using the home PC, voice recognition can be easily incorporated to allow the interface to be controlled by simple voice commands. The interface is designed using Visual Basic, which makes it a standard application that runs within Windows. The Basic Stamp 2 from Parallax interfaces with the home PC and with the various components and devices that will be controlled. Together these two devices offer a lot of power and control.

This document will explain all the components involved, including detailed descriptions and installation requirements. The system will be analysed with a demo system to illustrate the components working together. This will explain functionality, communication protocols, and programming specifics. The demo system is expanded into a complete system installed in a sample apartment. Pictures and wiring diagrams are provided to help the reader visualize the system’s functionality. Potential future upgrades are discussed, featuring upgrades that could
be implemented but were not tested. A cost analysis is provided illustrating the total price of
the complete system as it pertains to the sample apartment.

It is the author’s hope that after reading this document, the reader will have a thorough
understanding of how to build this particular system and be able to increase functionality to
create his or her own unique system.
2.0 Components

The components of the voice controlled home automation system allow the system to perform tasks. A component is an individual part that contributes to the larger entity, being the home automation system. The components listed are as follows:

- The home PC
- The interface
- The SayNow software
- The voice tracker array microphone
- The SoundMax sound beam array microphone
- The basic stamp2
- The stamp interface box
- The Sharp 8A solid state relay
- The PIR sensor
- The Sensiron temperature/humidity sensor
- The CRYDOM proportional power controller
- The Parallax (Futaba) Continuous Rotation Servo
- The line pulley mechanism
- The turn mechanism

Each component can be removed or a new component may be added. In this way, the system is fully upgradable and can be built over a long period while still allowing other components to function. A detailed list is provided to explain each component. Each component has a
complete description (including a picture), its purpose in the system is explained, and installation requirements are provided (including wiring diagrams).

2.1 The Home PC

![Image of a home PC]

Figure 1 The home PC

2.1.1 Description

The home PC is a personal computer designed to run commercial operating systems (ex: Windows, Linux). The home PC is configured as follows:

- Operating system: Windows XP Home Edition Service pack 3
- Processor: AMD Athlon 64 X2 Dual Core Processor 4400+ 2.32 Ghz
- RAM: 4 GB
- Hard Drives: 1TB, 500 GB, and 80GB

The technical specification for this device can be very flexible, but the minimum requirements would require:

- Operating system: Windows XP any version
- Processor: Pentium 4 1.8Ghz Processor or better
- RAM: 512 MB
- Hard drive: 3GB of free hard disk space
- Additional: Six USB 2.0 ports.

2.1.2 Purpose in the system

The purpose of this home PC is to allow a user-friendly interface to the system. Much advancement has been made in voice recognition technology, which allows many versions of free software to be implemented in this system. The PC also easily integrates other components to be attached that all rely on the PC for instruction.

2.1.3 Installation

In this system the PC is installed in the living room because of the typical components found in a living room (i.e. T.V.), and this is the room where many people spend most of their time.
2.2 The Interface

![Image of the interface](image)

2.2.1 Description

The Interface is software designed in Visual Basic by the author. The software is used on the home PC and can be controlled using the SayNow software or a keyboard and mouse. It will display the current room temperature, the room humidity, the output to the heating element, the status of the house, the day, the date, the time, alarm status, and visual real time graphs of the temperature, set point, and output. The user can push the command buttons, adjust the temperature set point, set on and off times, set various alarm conditions, and view a video feed provided by the webcam.

2.2.2 Purpose in the system

The Interface is the control center for the home automation system. The user, either directly or indirectly, must interact with this software to have the system perform functions. The interface
uses the home PC to interact with the Stamp 1, 2, 3 and 4. It determines which light or blind the user wishes on/open or off/closed based on the user’s input using the Stamp 1 Protocol. The interface reads and interprets the temperature from Stamp 2 based on the Stamp 2 Protocol. The interface reads motion and alarm status from the Stamp 3 using the stamp 3 protocol. The interface sends the appropriate output for the heating element to the Stamp 4 using the Stamp 4 Protocol. The interface performs a PID algorithm to control the temperature of a given environment.

2.2.3 Installation

The Interface is installed on the home PC using the Interface setup.exe file.

2.3 The SayNow Software

![Figure 3 The SayNow interface](image)

2.3.1 Description

The SayNow software is voice recognition software based on the Microsoft SAPI (speech API) and the .Net framework. The software is very small and uses few resources to run. The software has a voice training function to increase its recognition ability and allow added commands.
2.3.2 Purpose in the system

The SayNow software’s purpose in the system is to interpret voice commands from the user and control the interface based on those commands.

2.3.3 Installation

The SayNow software is installed on the home PC using the SayNow setup.exe

2.4 The Voice Tracker Array Microphone

![Voice tracker array microphone](1)

2.4.1 Description

The Voice Tracker Array microphone is an 8-microphone array that automatically locates an active speaker and electronically steers a listening beam in that direction. The microphone improves signal to noise ratio in two ways. Its digital signal processor creates a listening beam that will focus on a person speaking and filters noise from all other directions. Its proprietary noise reduction algorithms filter background noise and reverberations present in an acoustic environment. The microphone has a certified range of two feet, but for this project, the author has found 100% accuracy at six feet with natural speaking tone. Its dimensions are 18 x 2.5 inches (1).
2.4.2 Purpose in the system

The purpose of the Voice Tracker Array Microphone is to be the main input device for voice commands. It will relay the commands to the Home PC for processing and the Say Now Software will interpret the command and provide the appropriate action in the Interface.

2.4.3 Installation

Voice Tracker Array Microphone’s will be installed in the living room, the bedroom, the office, and the kitchen. This is because these are very large rooms. They will be installed in a manner that offers the greatest coverage for the microphone to listen. The microphone must be either rested on a shelf or bolted to a wall at a close relation to the speaker’s height. The microphone will attach to the computer via the 3.5mm stereo in jack or ‘mic in’ jack on the home PC. The PC has only one ‘mic in’ jack so a splitter will be required. The device requires 6VDC from a wall outlet and uses 400mA while it is listening.

2.5 SoundMax SoundBeam Array Microphone

![SoundMax soundbeam array microphone](image)

2.5.1 Description

The SoundMax SoundBeam Array microphone is a two-microphone array. It is not as sophisticated as the Voice Tracker Array Microphone, but is a powerful device that is inexpensive. The device can reduce a number of interferences by focusing a short listening beam directly in front of it. Its dimensions are 5 x 3 inches.
2.5.2 Purpose in the system

The SoundMax SoundBeam Array Microphone is an input device that will receive voice commands. It is used to cover a small amount of space that the voice tracker’s listening beam doesn’t cover. Its use in the system is limited but necessary for a seamless voice command home automation system.

2.5.3 Installation

SoundMax SoundBeam Array Microphone’s will be installed at the main entrance into the apartment, in the hallway, and the bathroom. This is because these are small areas and are the least occupied spaces in the apartment. The microphone only requires a 3.5mm connection to the Home PC. Since there is only one 3.5mm Jack on the Home PC a splitter will be required.

2.6 The Basic Stamp2 Microcontroller

![Basic Stamp 2 microcontroller](image)

Figure 6 Basic Stamp 2 (2)

2.6.1 Description

The Basic Stamp2 Microcontroller is a small IC used in many electronics projects and typically serves as the ‘brains’. The Microcontroller has 16 I/O pins used to output or input low voltages (5VDC) digitally to interface with other components. It also has serial communication pins
necessary for communications with a PC. The Microcontroller is programmed using software supplied by Parallax called PBasic, a variation of the BASIC programming language. Four Stamp2s will be used in this system (2).

### 2.6.2 Purpose in the system

The Basic Stamp2's purpose in the system is to interface with the other devices in the system, the relays, servos, temperature sensor, PIR sensor, etc. It serves as the link to these devices and the home PC. Its general function is to relay information from devices to the PC, but it also provides basic arithmetic and data translation.

### 2.6.3 Installation

The Basic Stamp2s will be housed in the stamp interface box located behind the TV. The location is only limited to being at least three feet from the home PC and six feet from an AC wall outlet. The Basic Stamp2 is connected to the Home PC via USB serial port. Stamp 1 will connect to serial port 1 of the home pc. Stamp 2 is connected to serial port 2. Stamp 3 is connected to serial port 3. Stamp 4 is connected to serial port 4.
2.7 The Stamp Interface Box

Figure 7 Stamp interface box

2.7.1 Description

The interface box is a small wooden box that houses all the stamps. The box contains an extension outlet that the stamps can plug into to receive power. The outlet is attached to an extension cord to be plugged in. Four outlets are provided to supply optional power to any other component. Four holes on the bottom edge allow the stamp’s USB cords to exit the box and attach to the Home PC. The front of the box has various connection points labelled to guide the home automation installer to where each control wire is to be plugged in, relating to the interface. These points correlate to each stamp’s I/O pin, Vdd (5VDC) and Vss (ground). The unassigned points are unwired terminals present to allow future upgrades. Inside the interface box connected to LRP, KP, OP, and BP are voltage-conditioning circuits that allow these pins to output 2-10V from the Basic stamp 2. Refer to figure 8 for a voltage-conditioning circuit.
2.7.2 Purpose in the system

The stamp interface box’s purpose is to allow easy first time or moving installation.

2.7.3 Installation

The stamp interface box is to be mounted to a wall close to the Home PC and an available outlet. The power cords plug into an available outlet the USB cords will be inserted into the USB ports in the home PC. Refer to figure 9 for stamp I/O to interface box wiring.

Figure 8 Voltage-conditioning circuit

<table>
<thead>
<tr>
<th>Stamp 1</th>
<th>Stamp I/O</th>
<th>Stamp interface box</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>LRL</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>KL</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>HL</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>BRL</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>BL</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>SR</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>SL</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>SK</td>
<td></td>
</tr>
<tr>
<td>P11</td>
<td>TL1</td>
<td></td>
</tr>
<tr>
<td>P12</td>
<td>TR1</td>
<td></td>
</tr>
<tr>
<td>P13</td>
<td>TL2</td>
<td></td>
</tr>
<tr>
<td>P14</td>
<td>TR2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stamp 2</th>
<th>Stamp I/O</th>
<th>Stamp interface box</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>LRTD</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>LRTC</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>KTD</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>KTC</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>QTD</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>OTC</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>BTD</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>BTC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stamp 3</th>
<th>Stamp I/O</th>
<th>Stamp interface box</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>LRM</td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>LRM1</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>KM</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>OM</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>HM</td>
<td></td>
</tr>
</tbody>
</table>
2.8 The Sharp 8A Solid State Relay

![Sharp 8A solid-state relay](image)

2.8.1 Description

The Sharp 8A solid-state relay is a small inexpensive solid-state relay. It allows isolated control of an AC voltage from 1.2V DC. It can control up to 240V AC. Its dimensions are .73 x .22 x .97 inches.

2.8.2 Purpose in the system

The Sharp 8A solid-state relay is used to control the AC voltages delivered to the lights in the system. The Stamp can then digitally activate any light.
2.8.3 Installation

The Sharp 8A solid-state relay requires 1.2 V DC to control an AC load. The stamp pin output is 5V DC so a voltage divider circuit is required before the voltage is applied to the relay. The signal from the stamp is sent through the voltage divider (60 ohm in series with 190 ohms) to Vss or stamp ground. The relay pins C1 and C2 are connected in parallel with the 60 ohm resistor to acquire 1.2 V. C1 is connected on the most positive end of the resistor and C2 is connected to the most negative end. The relay powers the light by completing the neutral part of the circuit. Refer to Figure 11 for the wiring component diagram.

Figure 11 Sharp 8A solid-state relay component wiring
2.9 The PIR Sensor

![PIR Sensor](image)

Figure 12 PIR Sensor (2)

2.9.1 Description

The PIR Sensor is a passive infrared sensor that can detect changes in infrared light. This allows the PIR Sensor to become a motion detector. The sensor can detect motion up to 20 feet away. The dimensions are 1.27 x 0.96 x 1.0 inches. The sensor is a pyroelectric device, meaning it has elements made of a crystalline material that generates an electric charge when exposed to infrared radiation (2). When motion is detected it will output a +5VDC signal.

2.9.2 Purpose in the system

The PIR sensor is the motion detection element in the system. This element serves two purposes. The first is a method of activating lights, when the “lights” command is spoken. The second is for the security system, triggering an alarm during alarm mode. Eight PIR sensors will be used in this system.

2.9.3 Installation

The PIR sensor is connected and interfaced to the stamp. It requires an I/O pin, a 5VDC connection and ground from the Basic Stamp2. The sensor itself is mounted in a few key locations as follows:
- Two in the living room and the bedroom (one located in the middle of the room and one close to the entrance).
- One in every other room located close to the entrance

Refer to Figure 13 for the wiring component diagram.

![Figure 13 PIR sensor component wiring](image)

### 2.10 Sensiron Temperature/Humidity Sensor

![Figure 14 Sensiron temperature/humidity sensor (2)](image)

#### 2.10.1 Description

The sensiron temperature/humidity sensor is a smart sensor that measures temperature and humidity. The sensor communicates over a two wire serial interface. It provides many all in one functions. For instance, it has an analogue to digital interface built into it. This allows the sensor to be incredibly easy to use with the stamp that only accepts a digital signal. Considering it
measures temperature and humidity, it is worth the extra money as opposed to buying separate sensors for these variables. The sensor is factory calibrated. It can measure temperature within 0.01 degrees resolution and within +/-2 degree accuracy. It measures humidity with a resolution of 0.03% and within 3.5% accuracy. Its dimensions are 0.43 X 0.49 inches. It requires about 30 uW to run (2).

2.10.2 Purpose in the system
The Sensiron Temperature/Humidity Sensor’s purpose is to measure the temperature and humidity of a given room and relay that information to the stamp, which will then send it to the home PC when requested. In order to have PID control over the temperature a reliable and accurate temperature sensor is needed.

2.10.3 Installation
The Sensiron Temperature/Humidity Sensors requires 5VDC power connection from the Basic Stamp, a ground connection, and two I/O pins for serial communication. In this system, the sensor will be mounted close to where a person spends the majority of time in a given room. In the living room, it will be mounted close to the couch. Refer to Figure 15 for the wiring component diagram.

```
<table>
<thead>
<tr>
<th>Stamp I/O Pin</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stamp I/O Pin</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Clock</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vss</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 15 Sensiron temperature/humidity sensor component wiring
2.11 The CRYDOM Proportional Power Controller

![Figure 16 CRYDOM proportional power controller (3)](image)

2.11.1 Description

The CRYDOM proportional power controller will apply power proportionally to a load of 120/240VAC based on an input range of 2-10VDC. The controller is rated to deliver 15A on its output. Its dimensions are .91 x .91 x 1.8 inches.

2.11.2 Purpose in the system

The Proportional power controller is used to control the heating element. It delivers a power output that is proportional to the input signal. When the signal requests an output of 50% of power, the controller will apply 50% of power. In this way, the interface from the home pc can control the heating element.

2.11.3 Installation

The Proportional power controller’s 2-10 VDC delivered by a conditioned signal from the Stamp 4. The controller will complete the AC circuit by closing the neutral line. Refer to Figure 17 for the wiring component diagram.
2.12 The Parallax (Futaba) Continuous Rotation Servo

2.12.1 Description

The parallax Continuous Rotation Servo is a bidirectional servo that can reach a speed of 50RPM. The servo is controlled by the stamp through pulse width modulation. The servo’s dimensions are 2.2 x 0.8 x 1.6 inches. It requires 5VDC to run.

2.12.2 Purpose in the system

The parallax Continuous Rotation Servo is used to control the curtains and blinds in the apartment. Two Servos will be used to provide motion to the pulleys of the line pulley system.
mechanism in the living room and one servo will provide motion to the turn mechanism in the kitchen. Three of these servos will be used in this system.

2.12.3 Installation

The Servo will be connected to the line pulley mechanism in the living room and the turn mechanism in the kitchen. Each servo connects to a stamp I/O pin, the Vdd 5V supply and the Vss Ground connections on the Basic Stamp. Refer to Figure # for the component-wiring diagram.

![Figure 19 Servo component wiring](image)
2.14 The Line Pulley Mechanism

![Line pulley mechanism](image)

2.14.1 Description

The line pulley mechanism is made of two parts. Each part consists of two garage door pulleys placed at the opposite ends of a 45” plank of wood. A clothesline is connected around the pulleys.

2.14.2 Purpose in the system

The line pulley mechanism’s purpose is to allow the curtains to move smoothly.

2.14.3 Installation

The line pulley mechanism is mounted behind the top of the curtains. The left side of the mechanism is placed three inches higher than the right. The right end of the left curtain is connected to the bottom portion of the clothesline on the left half of the mechanism. The left end of the right curtain is connected to the top portion of the clothesline on the right half of the mechanism.
2.15 The Turn Mechanism

![Figure 21 Turn mechanism](image)

2.15.1 Description

The turn mechanism is made of a half square bracket that has a pin through it that fits through the blind shade open mechanism.

2.15.2 Purpose in the system

The turn mechanism is used to open the kitchen blinds.

2.15.3 Installation

The turn mechanism is connected to a The Parallax (Futaba) Continuous Rotation Servo and the kitchen blind. A half D bracket is used to stabilize the servo to the wall.
2.15 The LED Automatic Closet Light

![LED Automatic Closet Light](image)

Figure 22 LED automatic closet light (4)

2.15.1 Description

The LED automatic closet light consists of an LED, a magnet, and a magnetically controlled circuit. The circuit is normally closed. When the magnet is placed at least two inches from the circuit, it opens. The LED is powered by the circuit.

2.15.2 Purpose in the system

The LED automatic closet light has two purposes in the system. It is the alarms mentioned throughout the report and is the trip circuit for the line pulley mechanism.

2.15.3 Installation

The LED Automatic closet light used as the alarm or the trip circuit is wired the same way. A wire is connected across the anode of the LED to the appropriate I/O pin on the stamp. Another wire is connected to the cathode of the LED and ground. In this way as the magnet separates from the circuit a high signal of 3VDC is sent to the Basic Stamp 2. Refer to figure 23 for LED connection to stamp I/O.
Figure 23 Connection to LED
3.0 The Demo System

The purpose of the demo system is to show how the components come together to create a voice controlled home automation system. The demo system assumes no actual house installation but the programming of the stamps and interface is complete to work with the voice controlled home automation system. The demo system will control two lamps, a curtain, a blind, and a heating element. The demo system will have the components needed to control these devices based on the functions of the component’s section. Both microphones will be used for the voice control.

3.1 The Interface

The user can interact in many different ways with the demo system using the interface, by pressing command buttons, inputting desired temperature etc. The interface communicates with the Basic Stamp2’s to control the other components. The interface also performs the PID calculation to control the heating. A list of the functions is described refer to Figure 24 to correlate the function with the number.
3.1.1 Interface display functions

1) The process variable graph. This graph displays in real time the current temperature and the user defined set point with temperature on the Y-axis and time as the X-axis. The temperature colour is cyan while the set point colour is yellow.

2) The output graph. This graph displays in real time the percentage signal that is sent to the Stamp 4 to control the heating element.

3) The temperature display. This displays the current temperature of the space where the temperature humidity sensor is.

4) The Humidity Display. This displays the current relative humidity of the space where the temperature humidity sensor is.
5) The Quality Display. This gives a notification of the current room quality concerning
temperature and humidity. If the temperature is equal to the set point, it will read optimal
temperature. If the temperature is greater than 30 it will display hot and if it is less than 20 it
will display cold. It will hold this value for a second and after it will display the humidity quality.
If the relative humidity is 55%, it will display optimal humidity. If it is greater than 80%, it will
read Damp and if it is less than 35%, it will read Dry. For any other value that is not mentioned
it will read acceptable temperature and then acceptable humidity.

6) The user selected temperature set point box. The user, through voice or keyboard/mouse,
will select the desired temperature from this box. The box is a list with values the user can
select from, these being: 5, 10, 15, 17, 20, 22, 25, 27, 30, 32, and 35.

7) The light control box. The command buttons that will control the lighting are in the light
control frame. There is an on and an off state that is included for each room. The rooms are
Living room, Kitchen, office, hallway, bedroom, and bathroom. The illuminate and deluminate
commands are also here which will turn on or off all lights respectively. The lights on button will
activate the motion activated lights sequence and the lights off button will deactivate the
motion activated lights sequence.

8) The blind control box. In this box, two sets of blinds can be controlled: the living room blinds,
which activate the line pulley mechanism, and the kitchen blinds, which activate the turn
mechanism. There is a button for open and one for close for each one that will have the
mechanism perform uniformly.
9) The economy settings box. This box has many features. First, through the off time list the user can set a specific time to turn the economy settings on. Through the on time list, the user can specify at which time the settings will turn off. The time starts at 12:00:00 and can be increased in 30-minute intervals. The user then specifies AM or PM. This is a feature to be implemented when the user leaves the house to go to work or some other extended leave time. While the economy settings are active, the program forces the temperature set point to zero. This turns off the heat in the house and renders other functions of the home automation system useless. There are check boxes located underneath the On Time list. These check boxes include the following:

- Emergency Temp. This sets the set point to 15 as opposed to zero. Primarily used for wintertime when the temperature of the house could fall to low temperatures.

- Alarm. This activates the alarm function. When a door or window is opened or there is motion through the house or a fire alarm activates the system will play a .wav file of loud sirens indicating an alarm condition.

- Weekend Omit. This box omits the economy settings during Saturdays and Sundays. This is in place so the schedule does not need to be changed for people that never work weekends.

- Phantom mode. This box represents the control that simulates someone being home while the economy settings are active. At 12:00:00 PM, the Living room lights will turn on until 1:00:00 PM. At 2:00:00PM, the kitchen lights will turn on until 2:15:00 PM. At 2:17:00 PM, the Living room lights will turn on again. At 3:00:00 PM, the living room lights will turn off and the bedroom lights will turn on until 4:00:00PM.
The economy box also contains an override option box labelled *I’m Home*, which will deactivate economy mode and *OK* which will allow economy mode to continue.

10) The output display box. This displays the current output percent that the interface has calculated.

11) The house status display box. This box displays the status of the house. If the temperature falls below 15 degrees or above 30 degrees Celsius, it will display *Temperature problem*. If the Humidity is below 20% or above 90%, it will display *Humidity problem*. If an alarm other than the fire alarm is activated, it will display *compromised*. If the fire alarm is activated, it will display *fire*. At any other time, it will display *secure*.

12) The Date boxes. These boxes display the current day, date, and time from left to right respectively.

13) The security box. These boxes display the current state of alarms. If a window is opened, it will display which window is opened in the windows box. If there is motion, the motion display box will indicate where the motion is. If a fire alarm has been triggered the fire display box will display which alarm is activated. If a door opens, the door display box will indicate which door.

14) The security camera. This displays the video of the connected webcam.

3.1.2 Interface Communications

The communication between the interface and the stamp occurs in two different ways. When sending or receiving data to and from the stamp, the interface uses the event driven method (4). Using this method, the interface only accesses the communication buffer when data is present. This allows the interface to communicate more efficiently than polling (4) the buffer
for receiving data. Polling the buffer is a method where the program is constantly checking the buffer to see if data is available (4). The Basic Stamp2’s only communicate, in this system, using the polling method. When sending or receiving data to or from the stamp the communication line will operate on a 2400 Baud rate that contains eight data bits and one stop bit. No parity bit will be used. When using the event driven method the interface will receive data when there is at least two bytes in the buffer. It will input data two bytes at a time, and does not have a DTR setting.

3.1.3 The PID algorithm

The PID control algorithm performed by the interface is a set of calculations performed by the software to determine how much power to allow the heating element to use. The power the heater uses is directly related to how much it will heat. The heating element is an on or off device. Conventional thermostats use a dead band control to control the heating of an environment. In this way, the temperature of the room is allowed to fluctuate by a given range. The heating system in this system adjusts the power proportionate to the output signal that the heater is allowed to use. This will minimize temperature fluctuations and offer more sophisticated control in a given environment. The PID algorithm takes the value of temperature in the room and compares it to the desired temperature. If there is an error in the comparison, the output is adjusted to bring the temperature to the desired value. It does this through a series of calculations demonstrated in figure 25:
Richard McAlpine  
Voice Controlled Home Automation

processpercent = ((lblTemperature.Caption / 35) * 100) 
sp = ((listSetpoint.Text / 35) * 100) 
If economy > 0 Then 
  sp = 0 
End If 
If (economy > 0) And (chkET = True) Then 
  sp = ((15 / 35) * 100) 
End If 
gain = 30 
Rate = 10 
reset = 3 
PV = processpercent 
derivativeG = PV + (lastinput - PV) * (Rate / 60) 
lastinput = PV 
derivative = derivative + (derivativeG - derivative) * 10 / 60 
Output = (sp - derivative) * (gain / 100) + integral 
If Output > 100 Then 
  Output = 100 
End If 
If Output < 0 Then 
  Output = 0 
End If 
integral = integral - (integral - Output) * (reset / 60)

Figure 25 PID algorithm Visual Basic code

The PID algorithm has many variables. The *processpercent* variable takes the temperature and converts it into a range of 0-100% from 0-35 degrees Celsius respectively. The same thing is done in the *sp* variable except it is converting the desired temperature value. The *gain, rate, and reset* refer to the proportional, derivative, and integral actions respectively the values are chosen arbitrarily. The *PV* variable is the value that is equal to the *processpercent* variable. The *derivativeG* variable adds the current *PV* variable to the error remaining from the last temperature reading that is multiplied by the *rate* variable. This allows the derivative action to respond to the rate of change in the error as the output is adjusted. The *lastinput* variable holds the value of the temperature before the temperature was manipulated by the output. The *derivative* variable adjusts the *derivativeG* value so its impact appears more spread. This is because the computer is performing the calculation every microsecond. The derivative variable
allows the derivative effect to seem as if it is applied at a greater interval. The output is a result of the error (defined as the desired temperature minus the temperature value conditioned by the derivative action) multiplied by the proportional gain and added to the integral variable. The integral variable holds the integral value and is determined by the last integral action subtracted by the current output, which is again subtracted by the last integral action and multiplied by the reset variable. This allows the integral action to have a ramping effect that will minimize error gradually ensuring the output signal doesn’t allow the temperature to overshoot the desired temperature. It is important to note that through this algorithm the integral and derivative terms won’t greatly affect the output if the temperature is not changing due to the output. If the system is installed and the output is proportional to the error, then an error is occurring with the stamp 4/interface communication, the stamp 4, the Max 518 digital to analogue converter or the heating element itself. The statements “if output > 100 then output = 100” and “if output < 0 then output = 0” are to ensure the integral action doesn’t accumulate values that are greater than 100% or less than 0%, this is known as anti-reset windup.

3.2 Voice Commands
The ability to have voice commands is crucial to this system’s user friendliness. The voice commands work by assigning each command a hot key, or navigating around the page. The command Living room lights on is activated by speaking “Living room lights on”. When this command is spoken, the SayNow software will simulate the pressing of <alt v>. The result will be as if the user had used the mouse and clicked the Living room lights on command button. In this way, adding the voice command function to the home automation program is rather simple and easy to upgrade and modify. A list of the voice commands and functions are displayed in
A problem with a voice-controlled system is false positive. A false positive is a spoken command that wasn’t intended to be a command. To overcome this problem an ‘initiate command’ is created that must be spoken to initiate a listening mode. After this command is spoken, the SayNow software will listen for a command for five seconds. If a command is not heard, the software will stop listening until it hears the initiate command. This command can be anything the user chooses. The default command is *computer*.

### 3.3 Communication protocols

#### 3.3.1 The Stamp 1 Protocol

The stamp 1 protocol is the protocol that the interface uses to communicate with the Stamp 1 (the light and blind stamp). In the stamp, each light or blind is assigned a *device* number. The lights will turn on or off based on a *status* number, *status* = 1 is on/open, *status* = 0 is off/close.

In this way, the data stream the stamp reads will be a variable *device* and a variable *status*. In this configuration, the interface with the same variables will assign the appropriate numerical values to these variables to the stamp. Refer to figure 26 for interface to stamp message, and figure 27 for stamp receive message.

```vba
Private Sub cmdOLightsOn_Click()
' Prep comm port
    MSComm1.CommPort = 1 ' Use COM1
    MSComm1.Settings = "2400,N,8,1" ' set the comm settings
    MSComm1.DTREnable = False ' no need for DTR
    MSComm1.PortOpen = True ' prepare for data
' turn on the light
    device = 2 ' the wired pin for Office light
    Status = 1 ' the state of the light
    MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) ' To the stamp we go
    MSComm1.PortOpen = False ' close comm port
End Sub
```

*Figure 26 Interface to stamp 1*
3.3.2 The Stamp 2 Protocol

The stamp 2 protocol is the protocol that the stamp 2 uses to send the temperature and humidity readings to the home PC. After the temperature is determined by the stamp 2, it adds 200 to the value. If it is a humidity reading, it adds 300 to the value. The stamp sends the data two bytes at a time, a high byte and a low byte. When the stamp sends either of these values to the interface, the interface can determine if it is a temperature reading because it will be in the 200 range, likewise the humidity reading will be in the 300 range. The interface simply subtracts the previously added value and displays the remainder. The protocol can be configured to add ranges for different rooms, e.g. 400 for bedroom temperature and 500 for bedroom humidity. When the interface receives the data, it receives it as a string. It reassembles the information through a private function and determines the data. Refer to figure 28 for stamp to interface message, and figure 29 for interface receive message.

```
SERIN 16,16780,[WAIT(255),device,Status] 'request input from PC
```

```
tC= (tC / 10) + 200
SEROUT 16,16780,[tC.HIGHBYTE, tC.LOWBYTE]
GOSUB SHT.Measure.Temp
PAUSE 1000
```

```
Humidity:
rhTrue = (rhTrue / 10) + 300
SEROUT 16,16780,[rhTrue.HIGHBYTE, rhTrue.LOWBYTE]
GOSUB SHT.Measure.Humidity
```

Figure 27 Stamp 1 receive from interface

Figure 28 Stamp 2 to interface
3.3.3 The Stamp 3 Protocol

The stamp 3 protocol is the protocol used by the stamp 3 to send the status of motion and alarms to the home pc to be interpreted by the interface. The stamp 3 assigns a number to the status of each alarm. If the alarm is on or motion is detected the number will be even starting at 50. If the alarm is off or there is no motion the number is odd starting at 51. It sends the data in the same way as the Stamp 2 protocol, in two bytes. Refer to figure 30 for stamp to interface message, and figure 31 for interface receive message.

```vbnet
Private Sub MSComm2_OnComm()
    'Get values from temp/humid stamp
    If MSComm2.CommEvent = comEvReceive Then
        sData = MSComm2.Input
        sHighByte = Asc(Mid$(sData, 1, 1))
        sLowByte = Asc(Mid$(sData, 2, 1))
        standard = JoinHighLows(sHighByte, sLowByte)
        If standard < 300 Then
            GoTo temperature
        End If
        If standard >= 300 Then
            GoTo humidity
        End If
        GoTo notta
    End If
End Sub
```

Figure 29 Interface receive from stamp 2

```plaintext
3.3.3 The Stamp 3 Protocol

The stamp 3 protocol is the protocol used by the stamp 3 to send the status of motion and alarms to the home pc to be interpreted by the interface. The stamp 3 assigns a number to the status of each alarm. If the alarm is on or motion is detected the number will be even starting at 50. If the alarm is off or there is no motion the number is odd starting at 51. It sends the data in the same way as the Stamp 2 protocol, in two bytes. Refer to figure 30 for stamp to interface message, and figure 31 for interface receive message.

LivingRoomLights:
send = 50
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PASSE 500

Figure 30 Stamp 3 to interface
```
The Stamp 4 Protocol is the protocol used by the interface to send data to the Stamp 4. The interface sends two variables to the stamp 4, room and temp. Room is given a numerical value determining which heating device will be affected. Temp is a numerical value of 0-255 that allows the stamp to adjust the heating element to the percentage output value determined by the interface. Refer to figure 31 for interface to stamp message, and figure 32 for stamp receive message.

```
Private Sub MSComm3_OnComm()
  When motion stamp has data
  If MSComm3.ComEvent = comEVReceive Then
    mData = MSComm3.Input
    mHighByte = Asc(Mid$(mData, 1, 1))
    mLowByte = Asc(Mid$(mData, 2, 1))
    motion = JoinHighLow(mHighByte, mLowByte)
    If motion = 50 Then
      lblMotion.Caption = "Living Room"
    ElseIf motion = 52 Then
      lblMotion.Caption = "Kitchen"
    ElseIf motion = 54 Then
      lblMotion.Caption = "Office"
    ElseIf motion = 56 Then
      lblMotion.Caption = "Hallway"
    ElseIf motion = 58 Then
      lblMotion.Caption = "Bedroom"
    ElseIf motion = 60 Then
      lblMotion.Caption = "Bathroom"
    Else
      lblMotion.Caption = "No Motion"
  End If
End If
```

Figure 31 Interface receive from stamp 3

3.3.4 The Stamp 4 Protocol

The Stamp 4 Protocol is the protocol used by the interface to send data to the Stamp 4. The interface sends two variables to the stamp 4, room and temp. Room is given a numerical value determining which heating device will be affected. Temp is a numerical value of 0-255 that allows the stamp to adjust the heating element to the percentage output value determined by the interface. Refer to figure 31 for interface to stamp message, and figure 32 for stamp receive message.

```
'Prep comm port
  MSComm4.CommPort = 4
  MSComm4.Settings = "2400,N,8,1"
  MSComm4.DIREnable = False
  MSComm4.PortOpen = True
'send out output
  temp = ((Output / 35) * 255)
  room = 0
  MSComm4.Output = Chr$(255) & Chr$(temp) & Chr$(room)
  MSComm4.PortOpen = False
```

Figure 32 Interface to stamp 4
3.4 Light control

The user controls the lights through the interface. When a command button is pressed, the interface determines the correct action to take based on the specific command button. The interface uses the Stamp 1 protocol to send the desired action to stamp 1. The Stamp 1 will apply power or remove power from the relay attached to the light. The light then turns on/or off respectively. The lights can also be controlled through motion. When the command button *lights on* is pressed, the interface uses the Stamp 1 protocol to trigger a program loop in the Stamp 1, motion loop. The loop accepts the values provided by the stamp for motion. When there is motion, a value is sent through the interface using the Stamp 3 protocol from stamp 3, the interface sends a motion indicator over the Stamp 1 protocol. Normally the Stamp 1 neglects this signal but while in motion loop the stamp will activate the light sequence that corresponds to the motion detected. When no motion is detected, the light will turn off. However, all of the standard lighting functions will override the motion action. If the user presses the command button *living room lights off*, the light will not turn on due to motion. The motion loop will have no effect on the particular light. The *lights off* command button will take the stamp out of this motion loop, and the stamp will resume normally. All lights may be activated by using the *illuminate* command or all lights deactivated using the *deluminate* command.
3.5 Blind control

In the interface, the blinds interact in the same way as the lights. Pressing the appropriate command button will either open or close the blinds using the Stamp 1 protocol. The stamp will receive the command and start the sequence that will turn the servo in the appropriate direction. The servo interacts with the pulley line mechanism to open or close the living room blinds, or another servo will interact with the turn mechanism to open the kitchen blinds.

3.6 Temperature control

The temperature control is a two-stage process: acquiring and interpreting the current temperature and controlling the heating element.

3.6.1 Acquiring and interpreting the current temperature

The Sensiron temperature/humidity sensor will relay room temperature to the stamp2 over its serial connection. The stamp2 uses the stamp 2 protocol to send this information to the home PC. The interface will display the temperature for the user to view. The temperature is also converted in the program to a value of 0-100% representing the temperature in a range of 0-35 degrees Celsius.

3.6.2 Controlling the heating element

The user selects a desired temperature from the set point list box in the interface. That temperature is converted to a value of 0-100%. The interface uses those two values in its PID algorithm to determine the appropriate output in percentage that will have the room temperature match the desired temperature. The output percentage value is converted to a range of 0-255 to be sent to the Stamp 4 using the Stamp 4 protocol. The signal is conditioned through the voltage conditioning circuit from the stamp interface box. The Max 518 Digital to
Analogue convertor portion conditions the signal to a range of 0-5 Volts. The range is then sent through the op-amp portion of the circuit that doubles the voltage. This allows the proportional power controller to apply the percentage of power value originally calculated by the interface. This method of control allows the system to keep the room temperature at a constant value, minimizing the oscillations experienced by traditional thermostat temperature control.
3.7 The demo system wiring

The components used in the demo system are as follows:

- The four Basic Stamp 2’s
- Two Sharp 8A solid-state relays
- Two Parallax continuous rotation servo’s
- Half of the line pulley mechanism
- The turn mechanism
- The Sensiron temperature/humidity sensor
- The voltage conditioning circuit from the stamp interface box
- One CRYDOM proportional power controller
- Two PIR sensors
- Two trip circuits

Refer to figure 34 for a hook-up diagram.

Figure 34 Demo System Wiring
4.0 The Voice Controlled Home Automation System

The voice controlled home automation system is an expansion to the Demo System, which illustrates a more complete system being utilized in a sample apartment.

4.1 The Apartment

The apartment is a two bedroom with 715 sq feet of space. The apartment is equipped with a 15.5 x 10.8 ft living room, an 11 x 9 ft kitchen, a 12.5 x 8.2 ft bedroom (referred to in this section as the office), and an 11.7 x 12.9 ft bedroom. Refer to Figure 36.
4.2 The Control Wiring

The apartment uses all the components previously mentioned for the system. Refer to Figure 37 to see how the control wiring and components are placed.
4.2.1 Motion Sensors

Eight PIR sensors are mounted in the ceiling with a control wire from each attaching to the interface box, each PIR sensor will require a connection to the Vss3 and the Vdd3 of the interface box.

- Living room. Two PIR sensors are mounted one close to the main door and one in the centre of the room. Their signal pins will attach to LRM and LRM1 respectively.

- Kitchen. One PIR sensor will mount 4 feet from the entranceway toward the center. The signal pin will connect to KM.
- Office. One PIR sensor will mount 4 feet from the entranceway toward the center. The signal pin will connect to OM.
- Hallway. One PIR sensor will mount in the center. The signal pin will connect to HM.
- Bedroom. Two PIR sensors are mounted one close to the main door and one in the center of the room. Their signal pins will attach to BRM and BRM1 respectively.
- Bathroom. One PIR sensor will mount in the center. The signal pin will connect to BM.

4.2.2 Alarms

12 alarms will be implemented including a fire alarm. Each alarm requires a connection to Vss3 and Vdd3. The alarms are mounted as two for each window and one for each door. The door alarms are mounted to have the circuit mounted to the frame and the magnet on the door. The window alarms are mounted with the circuit and magnet on the center of the window (in case the window is smashed) and the other window alarm will have the circuit on the frame and the magnet on the window. On each window the alarms will be wired so as to allow if either one activates the same high signal will be delivered to the stamp. The signal pin is wired as follows:

- The door alarm is connected to LRD
- The kitchen door alarm is connected to KD
- Living room window alarm is connected to LRA
- Kitchen window alarm is connected to KA
- Office window alarm is connected to OA
- Bedroom window alarm is connected to BA.
- The fire alarm is connected to FA.
4.2.3 Blind control

The blind’s motion is controlled by the Futaba Servo. Three servos are involved; each servo is connected to its corresponding mechanism. The Line pulley mechanism has to be mounted so as the L servo is on the furthest left and the R servo is on the furthest right. The Turn mechanism needs the turn pin to be placed through the hook hole of the blind and the L bracket mounted to the wall next to it. The servos each need a connection to Vss1 and Vdd1. Each trip circuit needs a connection to Vss1 and Vdd1. The connection from the signal wires is as follows:

- The L servo is connected to SL
- The R servo is connected to SR
- The TL1 trip circuit is connected to TL1
- TR1 trip circuit is connected to TR1
- TL2 trip circuit is connected to TL2
- TR2 trip circuit is connected to TR2

4.2.4 Temperature Measurement

Each Sensiron Temperature/Humidity sensor is mounted on a wall close to where a person would spend most of their time in the given room. Optimal locations are provided in figure 27. Each sensor will require a connection to the Vss2 and Vdd2 connections on the interface box. These connections will be made according to the wiring of the sensor already provided. Each sensor has a clock pin and a data pin. The pin connection is as follows:
· The sensor for the living room is connected as data pin to LRTD and the clock pin to LRTC

· The sensor for the kitchen is connected as data pin to KTD and the clock pin to KTC

· The sensor for the office is connected as data pin to OTD and the clock pin to OTC

· The sensor for the bedroom is connected as data pin to BTD and the clock pin to BTC.

4.2.5 Temperature Control

The temperature is adjusted by the proportional power controller’s from an analogue signal from the stamp interface box. The negative end of the proportional power controller is connected to Vss4 of the Interface box. Each positive connection is as follows:

· The living room power controller is connected to LRP

· The Kitchen power controller is connected to KP

· The office power controller is connected to OP

· The bedroom power controller is connected to BP.

4.2.6 Lighting

The lights are activated by the Sharp 8 A solid state relay. From a control perspective, the lights are on by applying voltage to the relay. Each relay is mounted where the current light switch is in each room. Each relay is connected to the interface box labelled Vss1 from the negative connection on the relay named as C2. The positive end of each relay is connected as follows:

· The living room light relay will connect to LRL

· The kitchen light relay will connect to KL

· The office light relay will connect to OL
- The hallway light relay will connect to HL
- The bedroom light relay will connect to BRL
- The bathroom light relay will connect to BL.

Refer to Figure 38 for the light control wiring.

4.3 The AC wiring

Some components in the voice controlled home automation system require a standard 120V AC source. Refer to Figure 38 for the AC component wiring. Each proportional power controller will require a connection referring to its wiring in the components section. Each solid-state relay will also need a connection.

Figure 38 The apartment with AC wiring, light control wiring, and furniture
4.4 Microphones

The microphones are the hardware that gives this system its ability for voice commands. Positioning these microphones is extremely important in order to reduce unwanted feedback and have seamless voice recognition throughout the apartment. Refer to figure 39. The microphones are placed as follows:

- One voice tracker in the living room against the front wall facing out
- One soundmax array beside the door targeting the kitchen
- One voice tracker in the kitchen against the front wall facing out
- One voice tracker in the office beside the entrance facing the back wall
- One soundmax in the hallway facing the bedroom
- One voice tracker in the bedroom against the closet facing the back wall
- One soundmax in the bathroom against the bathroom front wall facing the back wall
Figure 39 The apartment with microphone placement

4.5 The Bigger Picture

The system is designed to allow full control of the apartment from any location within. In this way, the apartment becomes an entity, allowing the user to interact with the apartment as opposed to just living in it. The automated components in the apartment will increase comfort and efficiency. The alarms and motion detection allow greater security. The temperature control ensures optimal temperature despite the conditions of the environment. The design of the system allows it to be easily modified and upgraded. A series of transparent wiring diagrams are supplied in Appendix C.
5.0 Potential Future Upgrades

This system is designed to be fully upgradable. By adding more stamps and adjusting the interface code, more components may be added to allow more control.

5.1 Automatic magnetically locked doors

Automatic magnetically locked doors could be easily integrated into the system. To do so would require a complete rethinking of the door system. The door itself would be a solid sheet of steel, roughly the size of a standard door. Four sets of wheels would be connected to the bottom of the door allowing it to slide. These wheels would fit into a rail that is made into the floor; this will prevent the door from sliding off course. The doors will open via dual action pneumatic pistons. The pneumatic pistons would receive air by solenoid valves that are controlled by a Basic Stamp2. The doors would have a locking sequence when closed. The locking sequence would include having the stamp power a circuit that will activate an electromagnetic lock that locks the door. If the door were configured to close by entering the frame parallel to it, a large dead bolt style lock would be created. This would greatly enhance the security of any home. By integrating this door system with the home automation system, the doors could open via motion sensor, voice command, keypad, or even fingerprint or retinal scan. Any of these could be used independently or in combination.

5.2 Moving Security Cameras

The servos already described could be used to provide motion to any low load. The webcam for instance could be attached to a servo allowing it to rotate. If more than one servo is used and attached to motion-conditioned device the camera could move theoretically on any axis. This
would allow the camera to view any angle of whatever room it was stationed. This could allow the camera’s viewing angle to be controlled by voice command, joystick (wired into the stamp) or remote control (also wired to the stamp). There are many webcams available today that offer a facial recognition feature. This does not mean that it can necessarily distinguish one person’s face from another, but it can distinguish a face from any other object. By having the drivers communicate with the interface, the security camera could track the movement of a face as it moves around the room.

5.3 Additional Interface Capabilities

5.3.1 Sending alerts to the user

The interface could be programmed to send an automatic e-mail whenever a user-defined event has occurred. An example of how this may be useful is if an alarm has been triggered an e-mail could be sent to alert the user. The same functionality could be applied to a phone call as well: when an event occurs, the interface would connect a VOIP program in the home pc that could dial a specified number and relay a recorded message.

5.3.2 Redesigning the voice recognition feature

The SayNow software could be removed from this system and replaced by voice recognition code built into the interface. The SayNow software is constructed in the visual studio environment using Microsoft SAPI (free with Windows). The software is not designed for this system therefore provides functionality that this system does not require. Having the voice recognition built into the interface would allow more accurate and efficient control.
5.3.3 Remote Control

The interface could broadcast over the internet using a remote desktop application. In this way, the interface could be controlled from anywhere in the world. A secure connection would be desired so hosting a website would most likely be required and would have to be paid for.
6.0 Conclusion

The voice controlled home automation system is an original design that can be applied to any home. The component-based implementation of this system allows it to be upgraded, reconfigured, and easily assembled. The cost of this system is $2570.79, but the price is dependent upon how much control is desired. Figure 37 displays cost analyses of the components in building the proposed system.

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>home PC</td>
<td>1</td>
<td>$600.00</td>
<td>$600.00</td>
</tr>
<tr>
<td>Interface</td>
<td>1</td>
<td>Free</td>
<td>$0</td>
</tr>
<tr>
<td>SayNow software</td>
<td>1</td>
<td>Free</td>
<td>$0</td>
</tr>
<tr>
<td>Voice Tracker Array Microphone</td>
<td>4</td>
<td>$249.00</td>
<td>$996.00</td>
</tr>
<tr>
<td>SoundMax sound beam array microphone</td>
<td>3</td>
<td>$4.00</td>
<td>$12.00</td>
</tr>
<tr>
<td>Basic Stamp 2</td>
<td>4</td>
<td>$49.00</td>
<td>$196.00</td>
</tr>
<tr>
<td>Stamp interface box</td>
<td>1</td>
<td>$100.00</td>
<td>$100.00</td>
</tr>
<tr>
<td>Sharp 8A solid-state relay</td>
<td>6</td>
<td>$3.99</td>
<td>$23.94</td>
</tr>
<tr>
<td>PIR sensor</td>
<td>8</td>
<td>$9.99</td>
<td>$79.92</td>
</tr>
<tr>
<td>Sensiron temperature/humidity sensor</td>
<td>4</td>
<td>$35.99</td>
<td>$143.96</td>
</tr>
<tr>
<td>CRYDOM proportional power controller</td>
<td>4</td>
<td>$80.00</td>
<td>$320.00</td>
</tr>
<tr>
<td>Parallax (Futaba) Continuous Rotation Servo</td>
<td>3</td>
<td>$12.99</td>
<td>$38.97</td>
</tr>
<tr>
<td>Line pulley mechanism</td>
<td>1</td>
<td>$50.00</td>
<td>$50.00</td>
</tr>
<tr>
<td>Turn mechanism</td>
<td>1</td>
<td>$10.00</td>
<td>$10.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>$2570.79</strong></td>
</tr>
</tbody>
</table>
7.0 References


**Personal Reference**

Drew, Mike. St.Lawrence College Instrumentation Technician.
8.0 Glossary

**Anti-reset windup**- Anti-reset windup is a term that refers to an action that prevents an integrated term from accumulating past a specified range.

**Baud**- Baud or baud rate refers to the symbols per second rate over a transmission medium (5).

**Clock pin**- The clock pin refers to the pin on a controller that sets the clock speed that a serial device requires for communication with a controller.

**Data pin**- The data pin refers to the pin on a controller that receives or sends data typically to a serial device.

**Derivative**- Derivative refers to a measure of how a function changes as its input changes (5).

**DTR**- DTR is an abbreviation for Data Terminal Ready. The DTR is a control signal in an RS-232 serial communications cable that is transmitted between a computer and another device used for flow control (5).

**False positive**- False positive refers to a condition that is true but unintended to be true at a given instant.

**Hot key**- Hot key is a term used to describe a keyboard shortcut. It is a set of keys that invoke an operation within software (5).

**IC**- IC is an abbreviation for an integrated circuit.

**Integral**- Integral refers to sums of functions with respect to tagged partitions of an interval (5).
**I/O**- I/O is an abbreviation for input and output.

**Load**- The load refers to the device that is to be controlled.

**Parity**- Parity or parity bit refers to the parity of transmitted data for error detection. If the parity of the transmission protocol is set to even, a bit will be added if the sum of the transmitted data is odd. Odd parity is the opposite to even parity (5).

**PID**- PID is an abbreviation for Proportional Integral Derivative.

**Pin**- A pin refers to a connection to an IC.

**Polling**- Polling refers to actively sampling the status of an external device by a client program as a synchronous activity (5).

**Proportional**- Proportional refers to two or more values that are related by a multiple of one of the values (5).

**Pyroelectric**- Pyroelectricity is the ability of certain materials to generate an electric potential when the material is heated or cooled (5).

**Resistor**- A resistor is an electronic component that produces a voltage across its terminals. The voltage is proportional to the current passing through it (5).

**SAPI**- SAPI is an abbreviation for Speech Application Programming Interface. SAPI is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within Windows applications (5).
**Servo** – A servo is a control system that converts a small mechanical motion into one requiring much greater power (5).

**Signal pin**- The signal pin refers to the pin on a controller that receives or sends a control signal.

**Smart sensor**- A smart sensor is a sensor that provides measurement capture, signal conditioning, and signal processing in one device.

**Solenoid valves**- A solenoid valve is an electromechanical valve that opens or closes based on an electrical current.

**Solid-state relay**- A solid-state relay is an electronic switch that contains no moving parts. The relay is photo coupled meaning the control signal is isolated optically from the load (5).

**USB**- USB is an abbreviation for Universal Serial Bus. USB is a specification to establish communication between devices and a host controller (5).

**Variable**- A variable is a label that is referenced in a program that can be assigned and hold values.

**Wav**- A wav refers to a small audio software file that is typically used for various sound FX. E.g. A doorbell sound.
9.0 Appendices

Appendix A

Appendix A.1 Voice commands and functions

<table>
<thead>
<tr>
<th>Voice Command</th>
<th>Function</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>Enter listening mode</td>
<td>NA</td>
</tr>
<tr>
<td>Living room lights on</td>
<td>Turns on living room lights</td>
<td>Alt+V</td>
</tr>
<tr>
<td>Living room lights off</td>
<td>Turns off living room lights</td>
<td>Alt+G</td>
</tr>
<tr>
<td>Kitchen lights on</td>
<td>Turns on kitchen lights</td>
<td>Alt+K</td>
</tr>
<tr>
<td>Kitchen lights off</td>
<td>Turns off kitchen lights</td>
<td>Alt+C</td>
</tr>
<tr>
<td>Office lights on</td>
<td>Turns on office lights</td>
<td>Alt+O</td>
</tr>
<tr>
<td>Office lights off</td>
<td>Turns off office lights</td>
<td>Alt+F</td>
</tr>
<tr>
<td>Hallway lights off</td>
<td>Turns on hallway lights</td>
<td>Alt+H</td>
</tr>
<tr>
<td>Hallway lights off</td>
<td>Turns off hallway lights</td>
<td>Alt+Y</td>
</tr>
<tr>
<td>Bedroom lights on</td>
<td>Turns on bedroom lights</td>
<td>Alt+R</td>
</tr>
<tr>
<td>Bedroom lights off</td>
<td>Turns off bedroom lights</td>
<td>Alt+E</td>
</tr>
<tr>
<td>Bathroom lights on</td>
<td>Turns on bathroom lights</td>
<td>Alt+B</td>
</tr>
<tr>
<td>Bathroom lights off</td>
<td>Turns off bathroom lights</td>
<td>Alt+A</td>
</tr>
<tr>
<td>Lights on</td>
<td>Activates motion light control</td>
<td>Alt+L</td>
</tr>
<tr>
<td>Lights off</td>
<td>Deactivates motion light control</td>
<td>Alt+S</td>
</tr>
<tr>
<td>Illuminate</td>
<td>Turns on all lights</td>
<td>Alt+I</td>
</tr>
<tr>
<td>Deluminate</td>
<td>Turns off all lights</td>
<td>Alt+D</td>
</tr>
<tr>
<td>Living room blinds open</td>
<td>Opens living room blinds</td>
<td>Alt+Z</td>
</tr>
<tr>
<td>Living room blinds close</td>
<td>Closes living room blinds</td>
<td>Alt+X</td>
</tr>
<tr>
<td>Kitchen blinds open</td>
<td>Opens kitchen blinds</td>
<td>Alt+W</td>
</tr>
<tr>
<td>Kitchen blinds close</td>
<td>Closes kitchen blinds</td>
<td>Alt+U</td>
</tr>
<tr>
<td>Degrees 5</td>
<td>Sets the temperature to 5 degrees</td>
<td>Alt+T, ↑ (12X)</td>
</tr>
<tr>
<td>Degrees 10</td>
<td>Sets the temperature to 10 degrees</td>
<td>Alt+T, ↑ (12X), ↓</td>
</tr>
<tr>
<td>Degrees 15</td>
<td>Sets the temperature to 15 degrees</td>
<td>Alt+T, ↑ (12X), ↓ (2X)</td>
</tr>
<tr>
<td>Degrees 17</td>
<td>Sets the temperature to 17 degrees</td>
<td>Alt+T, ↑ (12X), ↓ (3X)</td>
</tr>
<tr>
<td>Degrees 20</td>
<td>Sets the temperature to 20 degrees</td>
<td>Alt+T, ↑ (12X), ↓ (4X)</td>
</tr>
<tr>
<td>Degrees 22</td>
<td>Sets the temperature to 22 degrees</td>
<td>Alt+T, ↑ (12X), ↓ (5X)</td>
</tr>
<tr>
<td>Degrees 25</td>
<td>Sets the temperature to 25 degrees</td>
<td>Alt+T, ↑ (12X), ↓ (6X)</td>
</tr>
<tr>
<td>Degrees 27</td>
<td>Sets the temperature to 27 degrees</td>
<td>Alt+T, ↑ (12X), ↓ (7X)</td>
</tr>
<tr>
<td>Degrees 30</td>
<td>Sets the temperature to 30 degrees</td>
<td>Alt+T, ↑ (12X), ↓ (8X)</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Key Sequence</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>Degrees 32</td>
<td>Sets the temperature to 32 degrees</td>
<td>Alt+T, ↑(12X), ↓(9X)</td>
</tr>
<tr>
<td>Degrees 35</td>
<td>Sets the temperature to 35 degrees</td>
<td>Alt+T, ↑(12X), ↓(10X)</td>
</tr>
<tr>
<td>Emergency temp</td>
<td>Sets temperature to 15 degrees during off time</td>
<td>Alt+T, TAB, SPACE</td>
</tr>
<tr>
<td>Alarm</td>
<td>Activates alarms during off time</td>
<td>Alt+T, TAB (2X), SPACE</td>
</tr>
<tr>
<td>Weekend omit</td>
<td>Allows off time schedule to not activate during weekends</td>
<td>Alt+T, TAB (3X), SPACE</td>
</tr>
<tr>
<td>Phantom</td>
<td>Activates phantom mode during off time</td>
<td>Alt+T, TAB (4X), SPACE</td>
</tr>
<tr>
<td>Off time twelve</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X)</td>
</tr>
<tr>
<td>Off time twelve thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(2X)</td>
</tr>
<tr>
<td>Off time one</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(3X)</td>
</tr>
<tr>
<td>Off time two</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(4X)</td>
</tr>
<tr>
<td>Off time two thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(5X)</td>
</tr>
<tr>
<td>Off time three</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(6X)</td>
</tr>
<tr>
<td>Off time three thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(7X)</td>
</tr>
<tr>
<td>Off time four</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(8X)</td>
</tr>
<tr>
<td>Off time four thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(9X)</td>
</tr>
<tr>
<td>Off time five</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(10X)</td>
</tr>
<tr>
<td>Off time five thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(11X)</td>
</tr>
<tr>
<td>Off time six</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(12X)</td>
</tr>
<tr>
<td>Off time six thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(13X)</td>
</tr>
<tr>
<td>Off time seven</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(14X)</td>
</tr>
<tr>
<td>Off time seven thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(15X)</td>
</tr>
<tr>
<td>Off time eight</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(16X)</td>
</tr>
<tr>
<td>Off time eight thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(17X)</td>
</tr>
<tr>
<td>Off time nine</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(18X)</td>
</tr>
<tr>
<td>Off time nine thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(19X)</td>
</tr>
<tr>
<td>Off time ten</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(20X)</td>
</tr>
<tr>
<td>Off time ten thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(21X)</td>
</tr>
<tr>
<td>Off time eleven</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑(24X), ↓(22X)</td>
</tr>
<tr>
<td>Time</td>
<td>Function Description</td>
<td>Keyboard Shortcuts</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Off time eleven thirty</td>
<td>Sets the specified off time</td>
<td>Alt+T, TAB (4X), ↑ (24X), ↓ (23X)</td>
</tr>
<tr>
<td>On time twelve</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X)</td>
</tr>
<tr>
<td>On time twelve thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓</td>
</tr>
<tr>
<td>On time one</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (2X)</td>
</tr>
<tr>
<td>On time one thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (3X)</td>
</tr>
<tr>
<td>On time two</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (4X)</td>
</tr>
<tr>
<td>On time two thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (5X)</td>
</tr>
<tr>
<td>On time three</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (6X)</td>
</tr>
<tr>
<td>On time three thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (7X)</td>
</tr>
<tr>
<td>On time four</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (8X)</td>
</tr>
<tr>
<td>On time four thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (9X)</td>
</tr>
<tr>
<td>On time five</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (10X)</td>
</tr>
<tr>
<td>On time five thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (11X)</td>
</tr>
<tr>
<td>On time six</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (12X)</td>
</tr>
<tr>
<td>On time six thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (13X)</td>
</tr>
<tr>
<td>On time seven</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (14X)</td>
</tr>
<tr>
<td>On time seven thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (15X)</td>
</tr>
<tr>
<td>On time eight</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (16X)</td>
</tr>
<tr>
<td>On time eight thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (17X)</td>
</tr>
<tr>
<td>On time nine</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (18X)</td>
</tr>
<tr>
<td>On time nine thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (19X)</td>
</tr>
<tr>
<td>On time ten</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (20X)</td>
</tr>
<tr>
<td>On time ten thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (21X)</td>
</tr>
<tr>
<td>On time eleven</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (22X)</td>
</tr>
<tr>
<td>On time eleven thirty</td>
<td>Sets the specified on time</td>
<td>Alt+T, TAB (5X), ↑ (24X), ↓ (23X)</td>
</tr>
<tr>
<td>Off time P.M</td>
<td>Sets the specified off time to PM</td>
<td>Alt+P, ←</td>
</tr>
<tr>
<td>Off time A.M</td>
<td>Sets the specified off time to AM</td>
<td>Alt+P, →</td>
</tr>
<tr>
<td>On time P.M</td>
<td>Sets the specified on time to PM</td>
<td>Alt+M, ←</td>
</tr>
<tr>
<td>On time A.M</td>
<td>Sets the specified on time to AM</td>
<td>Alt+M, →</td>
</tr>
<tr>
<td>I’m home</td>
<td>Deactivates off time schedule</td>
<td>Alt+J</td>
</tr>
<tr>
<td>Resume</td>
<td>Resumes off time schedule</td>
<td>Alt+Q</td>
</tr>
</tbody>
</table>

Figure 41 Voice commands and functions
Appendix B

Appendix B.1 Interface code

Option Explicit
Private Declare Function Playwave Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal lpszSoundName As String, ByVal uFlags As Long) As Long
Dim Time           As Date                'set time as variable of the date function
Dim Day            As Long                'set Day as variable for the Day
Dim Today          As Date                'set Today as variable of the date function
Dim Ready          As Long                'temperature sense stamp
Dim Will           As Long                'to temperature sense stamp2
Dim Status         As Long                'Status as pin status
Dim device         As Long                'device as pin number
Dim temp           As Long                'tells stamp the output value for temperature output stamp
Dim room           As Long                'to tell stamp to stay in output loop
Dim mData          As String              'string holding motion location
Dim mHighByte      As Long                'highbyte of motion string
Dim mLowByte       As Long                'low byte of motion string
Dim motion         As Long                'clear representation of motion
Dim sData          As String              'string holding temperature and humidity value
Dim sHighByte      As Long                'high byte of temperature and humidity string
Dim sLowByte       As Long                'low byte of temperature and humidity string
Dim standard       As Long                'clear representation of temperature and humidity
Dim dfilter        As Long                'filter that scales down derivative affect
Dim derivativeG    As Long                'the derivative action
Dim lastinput      As Long                'the value of the last temperature reading before the new one
Dim PV             As Long                'the temperature scaled into a percent, process variable
Dim gain           As Long                'the proportionate gain
Dim Rate           As Long                'the derivative gain
Dim reset          As Long                'the integral gain
Dim derivative     As Long                'the true derivative action including the filter
Dim Output         As Long                'the final output of temperature process including the PID action
Dim integral       As Long                'the integral action
Dim sp             As Long                'the process setpoint
Dim processpercent As Long                'the temperature scaled into a percent used to allow PV to work with graph
Dim x              As Integer             'the graph x scale
Dim y              As Integer             'the graph y scale
Dim pvgraph(100)   As Integer             'the process variable graph
Dim outgraph(100)  As Integer             'the output graph
Dim setP           As Integer             'the setpoint for the graph
Dim economy        As Long                'holds the state of the economy mode
Dim over           As Long                'holds the state of the override
Dim normal         As Long                'holds the state of the phantom mode
Dim alarm          As Long                'holds the state of the alarm mode
Dim alarmm         As Long                'holds the value os a potential alarm condition
Dim eco1           As Long
Dim eco2           As Long
Private Sub Form_Load()

' on form load open motion stamp when the buffer has data fire comm
MSComm3.RThreshold = 2                  'every 2 bytes
MSComm3.InputLen = 2                   'input 2 at a time
MSComm3.Settings = "2400,N,8,1"         'comm settings
MSComm3.DTREnable = False              'no need for DTR
MSComm3.CommPort = 3                   'use comm 3 for motion stamp
MSComm3.PortOpen = True
' Prep comm port
MSComm2.CommPort = 2                    'Use COM2
MSComm2.Settings = "2400,N,8,1"         'set the comm settings
MSComm2.DTREnable = False               'no need for DTR
MSComm2.PortOpen = True                 'prepare for data

' activate stamp 3
Ready = 1                                 'the program has started read temperature
Will = 0                                   'the state of the light
MSComm2.Output = Chr$(255) & Chr$(Ready) & Chr$(Will)     'To the stamp we go
MSComm2.PortOpen = False                   'close comm port

' on form load open temperature stamp when the buffer has data fire comm
MSComm2.RThreshold = 2  
MSComm2.InputLen = 2  
MSComm2.Settings = "2400,N,8,1"  
MSComm2.DTREnable = False  
MSComm2.CommPort = 2  
MSComm2.PortOpen = True

'prepare the process graph
Picture1.Cls
Picture1.ScaleMode = 3
Picture1.ScaleHeight = 105
Picture1.ScaleWidth = 100
Picture1.AutoRedraw = True
Picture1.ForeColor = vbCyan
Picture1.DrawStyle = 0
Picture1.DrawWidth = 2

'prepare the output graph
Picture2.Cls
Picture2.ScaleMode = 3
Picture2.ScaleHeight = 105
Picture2.ScaleWidth = 100
Picture2.AutoRedraw = True
Picture2.ForeColor = vbRed
Picture2.DrawStyle = 0
Picture2.DrawWidth = 2

'to reduce errors in calculations
lblTemperature.Caption = 0  
lblHumidity.Caption = 0  
lblMotion.Caption = "No Motion"  
lblFire.Caption = "Safe"  
lblWindow.Caption = "Shut"  
lblDoor.Caption = "Shut"  
lblStatus.Caption = "Secure"  
optOffTimeAM = True  
optOnTimeAM = True  
End Sub

'main timer for continuous function
Private Sub Timer1_Timer()

End Sub
economy = (economy + 1)
End If
If economy > 0 Then
    lblStatus.Caption = "Economy"
End If
If Time = (cboOnTime.Text) & (txtAMPMOnTime.Text) Then
    economy = 0
    over = 0
End If
If optImHome = True Then
    economy = 0
    over = 1
End If
If optUsual = True Then
    normal = 1
End If
If (over = 1) And (normal = 1) And (eco1 = 1) Then
    economy = 1
End If
If (economy > 0) And (chkPretend = True) Then
    phantom = 1
Else
    phantom = 0
End If
If (economy > 0) And (chkAlrm = True) Then
    alarm = 1
Else
    alarm = 0
End If
If (alarmm = 1) And (alarm = 1) Then
    lblStatus.Caption = "compromised"
End If
If (alarm = 1) And (lblStatus.Caption = "compromised") Then
    Timer2.Enabled = True
Else
    Timer2.Enabled = False
End If
If Time = ("12:00:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdLRLightsOn_Click() = True
End If
If Time = ("1:00:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdLRLightsOff_Click() = True
End If
If Time = ("2:00:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdKLightsOn_Click() = True
End If
If Time = ("2:15:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdKLightsOff_Click() = True
End If
If Time = ("2:17:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdLRLightsOn_Click() = True
End If
If Time = ("3:00:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdLRLightsOff_Click() = True
End If
If Time = ("3:00:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdBRLightsOn_Click() = True
End If
If Time = ("4:00:00 PM") And (phantom = 1) And (economy = 1) Then
    cmdBRLightsOff_Click() = True
End If
End If
'-------------------------------------PID LOOP------------------------------------------
'PID Loop for temperature control
    processpercent = ((lblTemperature.Caption / 35) * 100)    'turning the process into a percent value 0-35 is 0-100%
    sp = (listSetpoint.Text / 35) * 100                      'turning the setpoint into a percent value 0-35 is 0-100%
    If economy > 0 Then                                       'if economy mode is on kill heater
sp = 0
End If
If (economy > 0) And (chkET = True) Then  
  'if emergency temp is on ensure temp is 15 degrees
  sp = ((15 / 35) * 100)
End If
gain = 30
Rate = 10
reset = 3
PV = processpercent  'process variable
derivativeG = PV + (lastinput - PV) * (Rate / 60)  'derivative gain
lastinput = PV  'to keep track of the last input value
derivative = derivative + (derivativeG - derivative) * 10 / 60  'derivative action
Output = (sp - derivative) * (gain / 100) + integral  'output with proportional, derivative and integral contribution
If Output > 100 Then  'anti reset windup
  Output = 100
End If
If Output < 0 Then  'anti reset windup
  Output = 0
End If
integral = integral - (integral - Output) * (reset / 60)  'integral action
lblOutput.Caption = Output  'Display output in form

'---Temperature, Setpoint and Output Graph-----------------------------
if economy > 0 Then
  setP = 0
Else
  setP = ((listSetpoint.Text / 35) * 100)
End If
If (economy > 0) And (chkET = True) Then
  setP = ((15 / 35) * 100)
End If
Picture1.Cls
pvgraph(100) = PV
For x = 0 To 99
  pvgraph(x) = pvgraph(x + 1)
  Picture1.PSet (x, 100 - (pvgraph(x)))
Next x
setP = 0
End Sub
Private Sub MSComm2_OnComm()
'Get values from temp/humid stamp
If MSComm2.CommEvent = comEvReceive Then  
  'If the data is to receive then process it
  sData = MSComm2.Input  
  'get data
  sHighByte = Asc(Mid$(sData, 1, 1))  
  'get 1st byte
  sLowByte = Asc(Mid$(sData, 2, 1))  
  'get 2nd byte
  standard = JoinHighLows(sHighByte, sLowByte)  
  'join bytes together
If standard < 300 Then  
  'if data is in the 200 range it is a temperature reading
  GoTo temperature
End If
If standard >= 300 Then  
  'if data is in the 300 range than it is a humidity reading
  GoTo humidity
End If
End If
End Sub

'--------------------------------------- Temperature Reading-------------------------------
temperature:
standard = standard - 200        'to isolate temperature
lblTemperature.Caption = standard
'To set alert status of environmental conditions
If standard > 30 Then
    lblQuality.Caption = "Hot"
ElseIf standard < 20 Then
    lblQuality.Caption = "Cold"
ElseIf Standard = listSetpoint.Text Then
    lblQuality.Caption = "Optimal Temperature"
Else
    lblQuality.Caption = "Acceptable Temperature"
End If
'Prep comm port
MSComm4.CommPort = 4                              'Use COM4
MSComm4.Settings = "2400,N,8,1"                     'set the comm settings
MSComm4.DTREnable = False                          'no need for DTR
MSComm4.PortOpen = True                            'prepare for data
'send out output
temp = ((Output / 35) * 255)                           'convertoutput to 0-255 and send to stamp
room = 0                                               'which room to output to
MSComm4.Output = Chr$(255) & Chr$(temp) & Chr$(room)  'To the stamp we go
MSComm4.PortOpen = False                              'close comm port
GoTo notta
'-----------------------------------------------------Humidity Reading-----------------------------------------------------

humidity:
standard = standard - 300        'to isolate humidity
lblHumidity.Caption = standard
If standard > 80 Then
    lblQuality.Caption = "Damp"
ElseIf standard < 35 Then
    lblQuality.Caption = "Dry"
ElseIf Standard = 55 Then
    lblQuality.Caption = "Optimal Humidity"
Else
    lblQuality.Caption = "Acceptable Humidity"
End If
GoTo notta
notta:
End Sub
Private Sub MSComm3_OnComm()                                 'when there is motion or some kind of alarm
'when motion stamp has data
If MSComm3.CommEvent = comEvReceive Then
    'if the data is to receive then process it
    mData = MSComm3.Input                                    'get data
    mHighByte = Asc(Mid$(mData, 1, 1))                       'get 1st byte
    mLowByte = Asc(Mid$(mData, 2, 1))                        'get 2nd byte
    motion = JoinHighLow(mHighByte, mLowByte)                'join bytes together
If motion = 50 Then
    lblMotion.Caption = "Living Room"                        'there is motion in the living room
ElseIf motion = 52 Then
    lblMotion.Caption = "Kitchen"                            'there is motion in the kitchen
ElseIf motion = 54 Then
    lblMotion.Caption = "Office"                             'there is motion in the office
ElseIf motion = 56 Then
    lblMotion.Caption = "Hallway"                            'there is motion in the hallway
ElseIf motion = 58 Then
    lblMotion.Caption = "Bedroom"                            'there is motion in the bedroom
ElseIf motion = 60 Then
    lblMotion.Caption = "Bathroom"                           'there is motion in the bathroom
Else
    lblMotion.Caption = "No Motion"                          'there is no motion
End If
If motion = (50) Or (52) Or (54) Or (56) Or (58) Or (60) Then
    alarmm = 1
End If
'Prep comm port to send to stamp 1 with lights
MSComm1.CommPort = 1                                     'Use COM1
MSComm1.Settings = "2400,N,8,1"                          'set the comm settings
Private Sub cmdKBlindsClose_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'close the blind
device = 8 'the kitchen blind function
Status = 0 'close blind
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdKBlindsOpen_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'open Blind
device = 8 'the kitchen blind function
Status = 1 'open blind
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdLRBlindsClose_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'close blind
device = 6 'the living room blind function
Status = 0 'close blind
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdLRBlindsOpen_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn on the light
device = 6 'the living room blind function
Status = 1 'open blind
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdBathLightsOff_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn off the light
device = 5 'the wired pin for Bathroom light
Status = 0 'the state of the light
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
Private Sub cmdBathLightsOn_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn on the light
  device = 5 'the wired pin for Bathroom light
  Status = 1 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdBRLightsOn_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn on the light
  device = 4 'the wired pin for Bedroom light
  Status = 1 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdBRLightsOff_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn off the light
  device = 4 'the wired pin for Bedroom light
  Status = 0 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdHLightsOn_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn on the light
  device = 3 'the wired pin for Hallway light
  Status = 1 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdHLightsOff_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn off the light
  device = 3 'the wired pin for Hallway light
  Status = 0 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdKLightsOff_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn off the light
  device = 2 'the wired pin for Kitchen light
  Status = 0 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdKLightsOn_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
'turn on the light
  device = 2 'the wired pin for Kitchen light
  Status = 1 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdLightsOff_Click()
'Prep comm port
  MSComm1.CommPort = 1    'Use COM1
  MSComm1.Settings = "2400,N,8,1"   'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
Private Sub cmdKLightsOn_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn on the light
device = 1 'the wired pin for Kitchen light
Status = 1 'the state of the light
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdKLightsOff_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn off the light
device = 1 'the wired pin for Kitchen light
Status = 0 'the state of the light
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdLRLightsOff_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn off the light
device = 0 'the wired pin for livingroom light
Status = 0 'the state of the light
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdLRLightsOn_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn on the light
device = 0 'the wired pin for livingroom light
Status = 1 'the state of the light
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdOLightsOff_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn off the light
device = 2 'the wired pin for Office light
Status = 0 'the state of the light
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub

Private Sub cmdOLightsOn_Click()
'Prep comm port
MSComm1.CommPort = 1 'Use COM1
MSComm1.Settings = "2400,N,8,1" 'set the comm settings
MSComm1.DTREnable = False 'no need for DTR
MSComm1.PortOpen = True 'prepare for data
'turn on the light
device = 2 'the wired pin for Office light
Status = 1 'the state of the light
MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdIlluminate_Click()
'Prep comm port
  MSComm1.CommPort = 1 'Use COM1
  MSComm1.Settings = "2400,N,8,1" 'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
' turn on all lights
  device = 30 'Signal to turn on all lights
  Status = 0 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdDeluminate_Click()
'Prep comm port
  MSComm1.CommPort = 1 'Use COM1
  MSComm1.Settings = "2400,N,8,1" 'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn off all lights
  device = 31 'Signal to turn off all lights
  Status = 0 'the state of the light
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdLightsOn_Click()
'Prep comm port
  MSComm1.CommPort = 1 'Use COM1
  MSComm1.Settings = "2400,N,8,1" 'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn off all lights
  device = 20 'Signal to turn on motion ready sequence
  Status = 0 'just sent for fun and games
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub
Private Sub cmdLightsOff_Click()
'Prep comm port
  MSComm1.CommPort = 1 'Use COM1
  MSComm1.Settings = "2400,N,8,1" 'set the comm settings
  MSComm1.DTREnable = False 'no need for DTR
  MSComm1.PortOpen = True 'prepare for data
'turn off all lights
  device = 27 'Signal to turn off motion ready sequence
  Status = 0 'just sent for fun and games
  MSComm1.Output = Chr$(255) & Chr$(device) & Chr$(Status) 'To the stamp we go
  MSComm1.PortOpen = False 'close comm port
End Sub
Private Function JoinHighLows(lHighs As Long, lLows As Long) As Long
  JoinHighLows = (lHighs * &H100) Or lLows
End Function
Private Function JoinHighLow(lHigh As Long, lLow As Long) As Long
  JoinHighLow = (lHigh * &H100) Or lLow
End Function
Private Sub Timer2_Timer()
  Call Playwave("Alarm.WAV", 3)
End Sub

Appendix B.2 Stamp 1 Code
'
' {$STAMP BS2}
' {$PBASIC 2.5}
' {$PORT COM1}
'LIGHT, BLIND, AND TEMPERATURE CONTROL USING A VISUAL BASIC HMI
'By: Richard McAlpine
'DECLARING VARIABLES
counter VAR Word                   'counter for counting hehehe
device VAR Byte                    'what device to be activated
status VAR Byte                    'the status of the device
LR VAR Byte                        'override motion sensor while LRLights is on in livingroom
K VAR Byte                         'override motion sensor in Kitchen
O VAR Byte                         'override motion sensor in Office
HW VAR Byte                        'override motion sensor in Hallway
BR VAR Byte                        'override motion sensor in Bedroom
BA VAR Byte                        'override motion sensor in Bathroom

'PINS AND CONNECTED OUTPUTS
'pin0=Livingroom lights
'pin1=kitchen lights
'pin2=office lights
'pin3=hallway lights
'pin4=bedroom lights
'pin5=bathroom lights
'pin6=livingroom blind right
'pin7=livingroom blind left
'pin8=kitchen blind
'pin11=livingroom blind left close trigger
'pin12=livingroom blind right close trigger
'pin13=livingroom blind left open trigger
'pin14=livingroom blind right open trigger
'PIN DECLARATIONS
OUTPUT 0
OUTPUT 1
OUTPUT 2
OUTPUT 3
OUTPUT 4
OUTPUT 5
OUTPUT 6
OUTPUT 7
OUTPUT 8
INPUT 9
INPUT 10
INPUT 11
INPUT 12
INPUT 13
INPUT 14

'START OF PROGRAM
'Main operating loop
Main:
LR = 0                                'Override reset for motion light loop
K = 0                                  'Override reset
O = 0                                  'Override reset
HW = 0                                 'Override reset
BR = 0                                 'Override reset
BA = 0                                 'Override reset

'start of light and blind loop
Lightsandblinds:
    SERIN 16,16780, [WAIT(255), device, status]       'request input from PC
    IF device = 0 THEN Livingroomlights            'Go to light loop
    IF device = 1 THEN Kitchenlights               '
    IF device = 2 THEN OfficeLights                '
    IF device = 3 THEN HallwayLights               '
    IF device = 4 THEN BedroomLights               '
    IF device = 5 THEN BathroomLights               '
    IF device = 6 THEN LivingroomBlinds            'Go to blind loop
    IF device = 8 THEN KitchenBlinds               '
    IF device = 20 THEN Main2                      'goto loop with motion triggering
    IF device = 30 THEN AllLights                  'goto all lights on
    IF device = 31 THEN NoLights                   'goto all lights off
GOTO main

'End of light and blind loop
'START OF BLIND CONTROLS
'living room blinds
'living room blinds open
Livingroomblinds:
IF Status = 1 THEN Livinroomblindsopen 'go to open livingroom blind loop
IF Status = 0 THEN Livinroomblindsclose 'go to close livingroom blind loop
GOTO main
Livinroomblindsopen:                          'open left and right blinds
PULSOUT 6, 650                               'right blind clockwise
PULSOUT 7, 1000                               'left blind counter clockwise
IF IN14 = 0 THEN LOW 6                        'stop right blind
IF IN14 = 0 THEN LRBoL                        'continue left blind
IF IN13 = 0 THEN LOW 7                        'stop left blind
IF IN13 = 0 THEN LRBoR                        'continue right blind
GOTO Livinroomblindsopen

LRBoL:                                        'finish left blind if right blind is complete
PULSOUT 7, 1000                               'left blind counter clockwise
IF IN13 = 0 THEN LOW 7                        'stop left blind
IF IN13 = 0 THEN main
GOTO LRBoL                                    'finish right blind if left blind is complete

LRBoR:                                        'finish right blind if left blind is finish
PULSOUT 6, 650                               'right blind clockwise
IF IN14 = 0 THEN LOW 6                        'stop right blind
IF IN14 = 0 THEN main
GOTO LRBoR                                    'end of living room blinds open

Livinroomblindsclose:                         'close left and right blinds
PULSOUT 6, 1000                               'right blind counter clockwise
PULSOUT 7, 650                                'left blind clockwise
IF IN12 = 1 THEN LOW 6                        'stop right blind
IF IN12 = 1 THEN LRBcL                        'continue left blind
IF IN11 = 1 THEN LOW 7                        'stop left blind
IF IN11 = 1 THEN LRBcR                        'continue right blind
GOTO Livinroomblindsclose

LRBcL:                                        'finish left blind if right blind is complete
PULSOUT 7, 650                                'left blind clockwise
IF IN11 = 1 THEN LOW 7                        'stop left blind
IF IN11 = 1 THEN main
GOTO LRBcL                                    'finish right blind if left blind is complete

LRBcR:                                        'finish right blind if left blind is finish
PULSOUT 6, 1000                               'right blind counter clockwise
IF IN12 = 1 THEN LOW 6                        'stop right blind
IF IN12 = 1 THEN main
GOTO LRBcR                                    'end of living room blinds close

'End of living room blinds
'Kitchen blinds
kitchenblinds:
IF status = 1 THEN KBOpen                     'If status is on then open the blinds
IF status = 0 THEN KBClose                   'If status is off then close the blinds
GOTO kitchenblinds                           'repeat just incase

KBOpen:                                       'open the kitchen slot blinds
FOR counter = 1 TO 500                        'go for 5 sec
PULSOUT 8, 500                               'turn clockwise
NEXT
LOW 8
GOTO main

KBClose:                                      'close the kitchen slot blinds
FOR counter = 1 TO 1000                       'go for 5 sec
PULSOUT 8, 1000                              'turn counter clockwise
NEXT
LOW 8
GOTO main

'end of kitchen blinds

'END OF BLIND CONTROL
'START OF ALL LIGHTS SINGLE COMMAND
'all lights on - illuminate command
Allights:
HIGH 0
HIGH 1
HIGH 2
HIGH 3
HIGH 4
HIGH 5
GOTO main
'all lights off - deluminate command
Nolights:
LOW 0
LOW 1
LOW 2
LOW 3
LOW 4
LOW 5
GOTO main
'END OF ALL LIGHTS SINGLE COMMAND
'start of Standard light control activation
Livingroomlights:
IF Status = 1 THEN HIGH 0
IF Status = 0 THEN LOW 0
GOTO main
Kitchenlights:
IF Status = 1 THEN HIGH 1
IF Status = 0 THEN LOW 1
GOTO main
OfficeLights:
IF Status = 1 THEN HIGH 2
IF Status = 0 THEN LOW 2
GOTO main
HallwayLights:
IF Status = 1 THEN HIGH 3
IF Status = 0 THEN LOW 3
GOTO main
BedroomLights:
IF Status = 1 THEN HIGH 4
IF Status = 0 THEN LOW 4
GOTO main
BathroomLights:
IF Status = 1 THEN HIGH 5
IF Status = 0 THEN LOW 5
GOTO main
'end of Standard light control activation
'light control with Lights command On allowing the lights motion triggering
'Main secondary operating loop
Main2:
'start of light and blind loop
lightsandblinds2:
SERIN 16,16780,[WAIT(255),device,Status]
IF device = 0 THEN Livingroomlightss
IF device = 1 THEN Kitchenlightss
IF device = 2 THEN OfficeLights
IF device = 3 THEN HallwayLights
IF device = 5 THEN BedroomLights
IF device = 5 THEN BathroomLights
IF device = 6 THEN Livingroomblindss
IF device = 8 THEN kitchenblindss
IF device = 50 THEN Livingroomlightssm
IF device = 51 THEN Livingroomlightssm1
IF device = 52 THEN Kitchenlightssm
IF device = 53 THEN Kitchenlightssm1
IF device = 54 THEN OfficeLightsm
IF device = 55 THEN OfficeLightsm1
IF device = 56 THEN HallwayLightsm
IF device = 57 THEN HallwayLightsm1
IF device = 58 THEN Bedroomlightsm
IF device = 59 THEN Bedroomlightsm1
IF device = 60 THEN Bathroomlightsm
IF device = 61 THEN Bathroomlightsm1
IF device = 27 THEN Main
IF device = 30 THEN Alllightsm
IF device = 31 THEN Nolightsm
GOTO main2
'end of light and blind loop
'START OF ALL LIGHTS SINGLE COMMAND
'all lights on - illuminate command
Alllightsm:
HIGH 0
HIGH 1
HIGH 2
HIGH 3
HIGH 4
HIGH 5
GOTO main2
'all lights off - deluminate command
Nolightsm:
LOW 0
LOW 1
LOW 2
LOW 3
LOW 4
LOW 5
GOTO main2
'END OF ALL LIGHTS SINGLE COMMAND
'start of motion light control activation
Livingroomlightsm:
IF LR = 1 THEN main2
HIGH 0
GOTO main2
Livingroomlightsm1:
IF LR = 1 THEN main2
LOW 0
GOTO main2
Kitchenlightsm:
IF K = 1 THEN main2
HIGH 1
GOTO main2
Kitchenlightsm1:
IF K = 1 THEN main2
LOW 1
GOTO main2
Officelightsm:
IF O = 1 THEN main2
HIGH 2
GOTO main2
Officelightsm1:
IF O = 1 THEN main2
LOW 2
GOTO main2
Hallwaylightsm:
IF HW = 1 THEN main2
HIGH 3
GOTO main2
Hallwaylightsm1:
IF HW = 1 THEN main2
LOW 3
GOTO main2
Bedroomlightsm:
IF BR = 1 THEN main2
HIGH 4
GOTO main2
Bedroomlightsm1:
IF BR = 1 THEN main2
LOW 4
GOTO main2
Bathroomlights:
IF BA = 1 THEN main2
HIGH 5
GOTO main2
Bathroomlights1:
IF BA = 1 THEN main2
LOW 5
GOTO main2
'end of motion light control activation
'start of Standard light control activation
Livingroomlights:
IF Status = 1 THEN HIGH 0
IF Status = 0 THEN LOW 0
LR = 1
GOTO main2
Kitchenlights:
IF Status = 1 THEN HIGH 1
IF Status = 0 THEN LOW 1
K = 1
GOTO main2
Office lights:
IF Status = 1 THEN HIGH 2
IF Status = 0 THEN LOW 2
O = 1
GOTO main2
Hallwaylights:
IF Status = 1 THEN HIGH 3
IF Status = 0 THEN LOW 3
HW = 1
GOTO main2
Bedroomlights:
IF Status = 1 THEN HIGH 4
IF Status = 0 THEN LOW 4
BR = 1
GOTO main2
'Bathroom lights:
IF Status = 1 THEN HIGH 5
IF Status = 0 THEN LOW 5
BA = 1
GOTO main2
'end of Standard light control activation
'START OF BLIND CONTROLS
'living room blinds
'living room blinds open
Livingroomblinds:
IF Status = 1 THEN Livinroomblindsopens
IF Status = 0 THEN Livinroomblindscloses
GOTO main2
Livinroomblindsopens:  'open left and right blinds
PULSOUT 6, 500
PULSOUT 7, 1000
IF IN14 = 1 THEN LOW 6
IF IN14 = 1 THEN LRBoLs
IF IN13 = 1 THEN LOW 7
IF IN13 = 1 THEN LRBoRs
GOTO Livinroomblindsopen
LRBoLs:  'finish left blind if right blind is complete
PULSOUT 7, 1000
IF IN13 = 1 THEN LOW 7
IF IN13 = 1 THEN main2
GOTO LRBoL
LRBoRs:  'finish right blind if left blind is finish
PULSOUT 6, 500
IF IN14 = 1 THEN LOW 6
IF IN14 = 1 THEN main2
GOTO LRBoRs
'end of living room blinds open
'living room blinds close
Livinroomblinds: 'close left and right blinds
PULSOUT 6, 1000
PULSOUT 7, 500
IF IN12 = 1 THEN LOW 6
IF IN12 = 1 THEN LRBcLs
IF IN11 = 1 THEN LOW 7
IF IN11 = 1 THEN LRBcRs
GOTO Livinroomblinds:
LRBcLs: 'finish left blind if right blind is complete
PULSOUT 7, 500
IF IN11 = 1 THEN LOW 7
IF IN11 = 1 THEN main2
GOTO LRBcLs
LRBcRs: 'finish right blind if left blind is complete
PULSOUT 6, 1000
IF IN12 = 1 THEN LOW 6
IF IN12 = 1 THEN main2
GOTO LRBcRs
'end of living room blinds close
'end of living room blinds
'Kitchen blinds
kitchenblinds:
IF status = 1 THEN KBOPENs                                      'If status is on then open the blinds
IF status = 0 THEN KBCLOSEs                                     'if status is off then close the blinds
GOTO kitchenblinds:
KBOPENs:                                                        'open the kitchen slot blinds
FOR counter = 1 TO 500                                          'go for 5 sec
    PULSOUT 8, 500
NEXT
LOW 8
GOTO main2
KBCLOSEs:                                                       'close the kitchen slot blinds
FOR counter = 1 TO 500                                          'go for 5 sec
    PULSOUT 8, 1000
NEXT
LOW 8
GOTO main2
'end of kitchen blinds
'END OF BLIND CONTROL

Appendix B.3 Stamp 2 Code

' {$STAMP BS2}
' {$PBASIC 2.5}
' {$SPORT COM2}
' I/O Definitions
ShtData CON 1       ' bi-directional data
Clock CON 0
' Constants
ShtTemp CON %00011   ' read temperature
ShtHumi CON %00101   ' read humidity
ShtStatW CON %00110   ' status register write
ShtStatR CON %00111   ' status register read
ShtReset CON %11110   ' soft reset (wait 11 ms after)
Ack CON 0
NoAck CON 1
No CON 0
Yes CON 1
MoveTo CON 2         ' for DEBUG control
ClrRt CON 11         ' clear DEBUG line to right
DegSym CON 186       ' degrees symbol for DEBUG
' Variables
ioByte  VAR  Byte  ' data from/to SHT1x
ackBit  VAR  Bit   ' ack/nak from/to SHT1x
toDelay  VAR  Byte  ' timeout delay timer
timeOut  VAR  Bit   ' timeout status
soT  VAR  Word  ' temp counts from SHT1x
tC  VAR  Word  ' temp - celcius
tF  VAR  Word  ' temp - fahrenheit
soRH  VAR  Word  ' humidity counts from SHT1x
rhLin  VAR  Word  ' humidity; linearized
rhTrue  VAR  Word  ' humidity; temp compensated
status  VAR  Byte  ' SHT1x status byte
will  VAR  Byte
ready  VAR  Byte

'begin reding temperature and humidity for sensor 1
reset:
SERIN 16,16780,[WAIT(255),ready,will] 'request input from PC
IF ready = 1 THEN Initialize
GOTO reset
Initialize:
GOSUB SHT_Connection_Reset
PAUSE 5000
Main:
Temperature:
tC= (tC / 10) + 200
SEROUT 16,16780,[tC.HIGHBYTE, tC.LOWBYTE]
GOSUB SHT_Measure_Temp
PAUSE 1000
Humidity:
rhTrue = (rhTrue / 10) + 300
SEROUT 16,16780,[rhTrue.HIGHBYTE, rhTrue.LOWBYTE]
GOSUB SHT_Measure_Humidity
PAUSE 1000
GOTO Main
END
SHT_Connection_Reset:
SHIFTOUT ShtData, Clock, LSBFIRST, [$FFF9]
SHT_Start:
INPUT ShtData
LOW Clock
HIGH Clock
LOW ShtData
LOW Clock
HIGH Clock
INPUT ShtData
LOW Clock
RETURN
'measure temperature
SHT_Measure_Temp:
GOSUB SHT_Start
ioByte = ShtTemp  ' temperature command
GOSUB SHT_Write_Byte  ' send command
GOSUB SHT_Wait  ' wait until measurement done
ackBit = Ack
GOSUB SHT_Read_Byte  ' another read follows
soT.HIGHBYTE = ioByte
ackBit = NoAck
GOSUB SHT_Read_Byte  ' get MSB
soT.LOWBYTE = ioByte
' last read
tC = soT / 10 - 400  ' convert to tenths C
tF = soT ** 11796 - 400  ' convert to tenths F
RETURN
'measure humidity
SHT_Measure_Humidity:
GOSUB SHT_Start  ' alert device
ioByte = ShtHumi  ' humidity command
GOSUB SHT_Write_Byte  ' send command
GOSUB SHT_Wait  ' wait until measurement done
ackBit = Ack  ' another read follows
GOSUB SHT_Read_Byte  ' get MSB
soRH.HIGHBYTE = ioByte
ackBit = NoAck  ' last read
GOSUB SHT_Read_Byte  ' get LSB
soRH.LOWBYTE = ioByte
' linearize humidity
  rhLin = (soRH * 0.0405) - (soRH * 2 + 0.0000028) - 4
  
' for the BASIC Stamp:
  rhLin = (soRH * 0.0405) - (soRH * 0.004 * soRH * 0.0007) - 4
  
' Conversion factors are multiplied by 10 and then rounded to
' return tenths
  rhLin = (soRH ** 26542)
  rhLin = rhLin - ((soRH ** 3468) * (soRH ** 3468) + 50 / 100)
  rhLin = rhLin - 40
' temperature compensated humidity
  rhTrue = ((tC - 25) * (soRH * 0.00008 + 0.01) + rhLin
  
' Conversion factors are multiplied by 100 to improve accuracy and then
' rounded off.
  rhTrue = ((tC / 10 - 25) * (soRH ** 524 + 1) + (rhLin * 10)) + 5 / 10
RETURN
'sends "status"
SHT_Write_Status:
GOSUB SHT_Start  ' alert device
ioByte = ShtStatW  ' write to status reg command
GOSUB SHT_Write_Byte  ' send command
ioByte = status
GOSUB SHT_Write_Byte
RETURN
' returns "status"
SHT_Read_Status:
GOSUB SHT_Start  ' alert device
ioByte = ShtStatW  ' write to status reg command
GOSUB SHT_Read_Byte  ' send command
ackBit = NoAck  ' only one byte to read
GOSUB SHT_Read_Byte
RETURN
'sends "ioByte"
'returns "ackBit"
SHT_Write_Byte:
SHIFTOUT ShtData, Clock, MSBFIRST, [ioByte]  ' send byte
SHIFTIN ShtData, Clock, LSBPRE, [ackBit\1]  ' get ack bit
RETURN
'returns "ioByte"
'sends "ackBit"
SHT_Read_Byte:
SHIFTIN ShtData, Clock, MSBFIRST, [ioByte]  ' get byte
SHIFTOUT ShtData, Clock, LSBFIRST, [ackBit\1]  ' send ack bit
INPUT ShtData  ' release data line
RETURN
' wait for device to finish measurement (pulls data line low)
' -- timeout after ~1/4 second
SHT_Wait:
INPUT ShtData  ' data line is input
FOR toDelay = 1 TO 250  ' give ~1/4 second to finish
timeOut = INS.LOWBIT(ShtData)  ' scan data line
IF (timeOut = No) THEN SHT_Wait_Done  ' if low, we're done
PAUSE 1
NEXT
SHT_Wait_Done:
RETURN
' reset SHT1x with soft reset
Appendix B.4 Stamp 3 Code

' {$STAMP BS2}
' {$PBASIC 2.5}
' {$PORT COM3}
'declare variables
lrm PIN 0
lrm2 PIN 1
km PIN 2
om PIN 3
hm PIN 4
bm PIN 5
bm2 PIN 6
br PIN 7
Send VAR Word
counter VAR Word
'pin0 = livingroom motion
'pin1 = livingroom motion
'pin2 = kitchen motion
'pin3 = office motion
'pin4 = hallway motion
'pin5 = bedroom motion
'pin6 = bedroom motion
'pin7 = bathroom motion
'pin8 =
'pin9 =
'pin10 =
'pin11 =
'pin12 =
'pin13 =
'pin14 =
'pin15 =
Begin:
FOR counter = 60 TO 0                   ' Wait 40 Seconds For PIR Warm-Up
 PAUSE 1000                          ' Display Counter Every Second
 NEXT
counter = 0                             ' Clear Counter Variable
Main:
 a:
 IF lrm2 = 1 THEN lrm = 1
 IF lrm = 1 THEN Livingroomlightsm
 b:
 IF lrm = 0 THEN Livingroomlightsn
 c:
 IF km = 1 THEN Kitchenlightsm
 d:
 IF km = 0 THEN Kitchenlightsn
 e:
 IF om = 1 THEN Officelightsm
 f:
 IF om = 0 THEN Officelightsn
 g:
 IF hm = 1 THEN Hallwaylightsm
 h:
 IF hm = 0 THEN Hallwaylightsn
 i:
 IF bm2 = 1 THEN bm = 1
 IF bm = 1 THEN Bedroomlightsm

IF bm = 0 THEN Bedroomlightsn
k:
IF br = 1 THEN Bathroomlightsm
l:
IF br = 0 THEN Bathroomlightsn
GOTO Main
Livingroomlightsm:
send = 50
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO b
Livingroomlightsn:
send = 51
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO c
Kitchenlightsm:
send = 52
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO d
Kitchenlightsn:
send = 53
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO e
Officelightsm:
send = 54
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO f
Officelightsn:
send = 55
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO g
Hallwaylightsm:
send = 56
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO h
Hallwaylightsn:
send = 57
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO i
Bedroomlightsm:
send = 58
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO j
Bedroomlightsn:
send = 59
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO k
Bathroomlightsm:
send = 60
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO l
Bathroomlightsn:
send = 61
SEROUT 16,16780,[Send.HIGHBYTE, Send.LOWBYTE]
PAUSE 500
GOTO a
Appendix B.5 Stamp 4 Code

' {$STAMP BS2}
' {$PBASIC 2.5}
' {$PORT COM4}
temp VAR Byte
room VAR Byte
channel VAR Byte
OUTPUT 0
OUTPUT 1
Main:
channel=0
SERIN 16,16780,[WAIT(255),temp,room] 'request input from PC
IF room = 0 THEN livingroom
livingroom:'DtoA protocol
OUT0=1
OUT1=1
PAUSE 500
OUT0=0
SHIFTOUT 0, 1, MSBFIRST, [%01011000\0\1,channel\0\1,\temp\0\1] 'output the 0 to 100% represented as 0 to 255
OUT1=1
GOTO Main

Appendix C

Appendix C.1 Wiring transparencies
Figure 42 Microphone placement
Figure 43 Furniture placement
Figure 44 Light control wire placement
Figure 45 AC wiring placement
Figure 46 Control wiring placement
Figure 47 Component placement
Figure 48 Apartment