Development
of a
Wireless Sensor Network

Swarthmore College
Department of Engineering
Senior Engineering Design Project

May 4, 2006

Masabumi Chano

Advisor: Dr. Lynne A. Molter

Abstract

A wireless sensor network, comprising of infrared and pressure sensors, was developed to monitor activity in a room and carry out particular tasks related to the activity monitored. One application was the monitoring of an entrance way to a room, in which lights were turned on or off according to the movement of people. The project was undertaken to explore sensor networks as a means to provide convenience to the user and aid in energy conservation. Various communication techniques were investigated using a communications trainer called the Telecommunications Instructional Modeling System for the transmission of information from the sensors to a control station. The control station, programmed in VHDL, interpreted the signals from the sensors properly and activated or deactivated corresponding tasks or appliances.

Table of Contents

1. Introduction and Background

This project follows in the footsteps of a previous work done by Anteneh Tesfaye, ’03, and David A. Whitehead, ’03, titled “Function-Specific Sensor Fusion”. In their project, David and Anteneh built a wired sensor network system that performed particular actions depending on the action detected by various sensors. The aim of this project is to test and investigate wireless communication techniques that could be implemented to develop a wireless sensor network. Heavy emphasis will be placed on testing different methods of wireless communication and see if clean transmission is observed. Since there are many ways in which a signal can be encoded and modulated, this project will explore different modes of communication to find ones that work well for sensor network applications.

The encoding, modulating, and transmission of the signals will be performed through a communications modeling system called the Telecommunications Instructional Modeling System (TIMS) from the Emona Instruments Pty. Limited. In effect, a wireless channel will be simulated using the TIMS.

Some of the motivations that inspired the project are convenience and energy consumption consciousness. For example, if a person were to walk into a dark room with his or her hands full, it would be convenient if the light automatically turned on upon detection of the person. Similarly, as humans, we are all likely to forget turning off appliances. If a sensor network was capable of monitoring rooms then, it could automatically turn off lights or other appliances when nobody is present in a room, and thus conserving energy. Although not discussed in this project, enhanced security

features could be implemented to work in conjunction with a sensor network to turn on or off lights to make a room appear active.

2. Theory

Many different wireless communication techniques were examined in this project. Some of the techniques examined are: Amplitude Shift Keying, Frequency Shift Keying, Binary Phase Shift Keying, Quaternary Phase Shift Keying, Pulse Code Modulation, Code Division Multiple Access, and Frequency Hop Spread Spectrum.

The first group of communication technique is called digital bandpass transmission. Here, digital signals can be used to modulate the amplitude, frequency, or phase of a carrier wave. If the digital signal is composed of rectangular pulses, then the resulting signal will be switched from one discrete value to another (Carlson, Crilly, & Rutledge, 2002). The second group of methods involves spread spectrum techniques such as frequency hop spread spectrum and code division multiple access.

Digital Bandpass Transmission

Modulated bandpass signals take the form,

$$x_c(t) = A_c\left[x_i(t)\cos(\omega_c t + \theta) - x_q(t)\sin(\omega_c t + \theta)\right]$$

The carrier frequency, amplitude, and phase in the above equation are all kept constant. However, the time-varying *i* (in-phase) and *q* (quadrature) components contain the message of the signal being transmitted.

Amplitude Shift Keying

Amplitude shift keying (ASK) is a form of amplitude modulation, in which digital data is expressed by variation in amplitude. The components of the equation $x_c(t)$ for describing ASK signal have the following characteristics:

$$x_i(t) = \sum_k a_k p_D(t - kD)$$
$$a_k = 0, 1, ..., M-1$$

The quadrature component of $x_c(t)$ is equal to zero since ASK signals do not undergo phase reversal. $x_i(t)$ is a unipolar non-return to zero (NRZ) signal. NRZ means a logic 1 bit is sent as a high value and a logic zero is sent as a low value. *M-1* defines the number of discrete amplitude variations in the ASK signal excluding an amplitude of 0. $p_D(t-kD)$ simply defines the starting point of the unipolar rectangular pulses.

In this project, binary ASK was examined, where a binary one was represented by the presence of a carrier wave and a binary zero was represented by the absence of a carrier. A simple way of producing such a signal could be done by turning off or on the carrier, a process known as on-off keying. For example, if the input is zero, the carrier will be turned off, producing a flat signal at 0V. In the case the input is one, the carrier will be on, and the signal will have a carrier frequency.

Another simple way of producing this signal can be done by multiplying the input signal with a carrier if the signals are binary. Thus, a zero will produce amplitude zero, while a one would give amplitude A_c, where A_c is the amplitude of the carrier.

Phase Shift Keying

Phase shift keying involves phase shifts to convey information regarding the input signal. For example, binary PSK waveforms contain phase shifts of $\pm \pi$ radians. Quarterary PSK would contain phase shift increments of $\pi/2$ radian. An easy way to observe this is to look at the constellation diagrams in Figure 2.

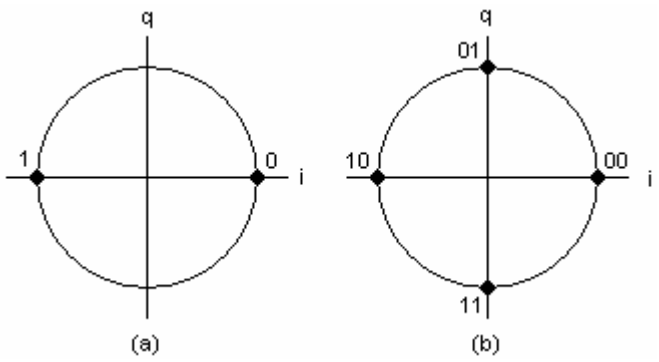

Figure 1. Constellation Diagrams. (a) BPSK, (b) QPSK

BPSK can be used to code 1 bit by differentiating the 0 and 1 by phase shifts of $\pm \pi$ radians. QPSK, on the other hand, can encode 2 bits by using phase shifts of multiples of $\pm \pi/2$ radians.

PSK signals can be expressed in general by the equations,

$$x_c(t) = A_c \sum_k \cos(\omega_c t + \theta + \phi_k) p_D(t - kD)$$

$$x_i(t) = \sum_k I_k p_D(t - kD)$$

$$x_q(t) = \sum_k Q_k p_D(t - kD)$$

$$I_k = \cos(\phi_k)$$

$$Q_k = \sin(\phi_k)$$

In order for there to be the largest possible phase modulation for a given value of M, the relationship between φ_k and a_k can be defined as follows:

$$\phi_k = \pi(2a_k + N)/M$$

A PSK signal with $M = 2$ is BPSK and $M = 4$ is QPSK.

Frequency Shift Keying

Frequency shift keying is a form of frequency modulation, in which digital data is modulated at different frequencies. One can imagine a switch selecting the modulating

signal different frequency generators. Conventional frequency shift keying can be thought of as the input signal controlling a switch that selects the modulated frequency from a bank of M oscillators. This process can be seen in block diagram form in Figure 2. However, this leads to discontinuities in the signal every time there is a switch in frequency. This will cause the resultant signal to have very large sidelobes which simply add bandwidth without carrying additional information (Carlson et al., 2002).

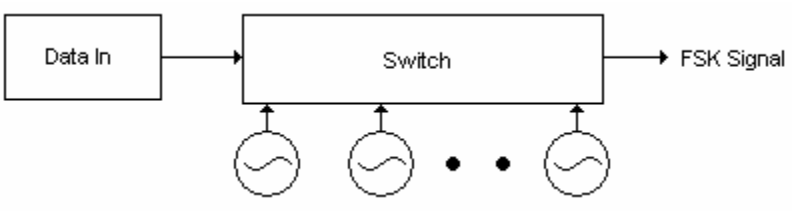

Figure 2. FSK generation using a switch

To avoid unnecessary sidelobes and discontinuities, one can implement a continuous phase FSK (CPFSK). Instead of the input signal selecting the modulating signal, it can be used to modulate the frequency of a single oscillator as shown in Figure 3. A sample oscilloscope output of a CPFSK signal can also be found below.

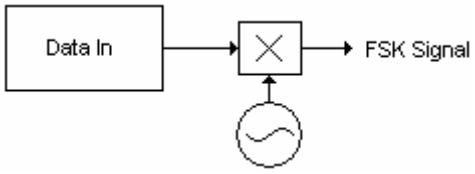

Figure 3. CPFSK generation

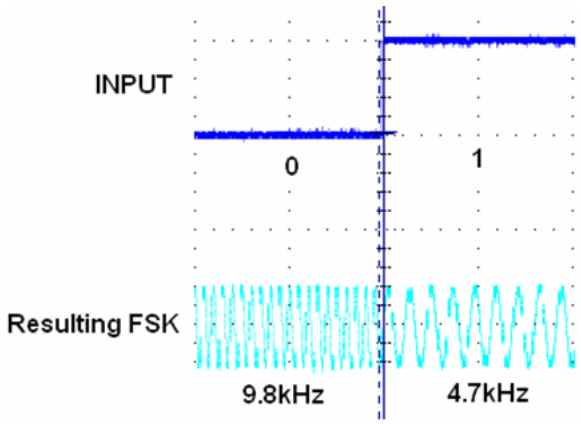

Figure 4. Original message and CPFSK modulated message

When the signal coming in is a binary zero, then the resulting signal has a carrier frequency of 9.8 kHz. When the signal is a binary one, then the resulting signal has a carrier of 4.7 kHz.

Spread Spectrum

Spread spectrum techniques involve the use of bandwidths far greater than the information bandwidth. The reason for this is to combat noise, interference, and unauthorized interception of message signals. Conventional transmission typically involved the use of known frequencies that could be easily detected or jammed. By using a spread spectrum technique, the signal would hop or change from one frequency to another in a pseudo-random fashion that is only known to the sender and receiver. The effect is enhanced security as well as resistance to noise and interference. Although spread spectrum techniques utilize a wider bandwidth than the information bandwidth, they are used effectively by allowing multiple users to share a wide frequency band.

Before explaining spread spectrum techniques, it is necessary to go over an important element in the generation of such signals. Pseudo-random sequences are sets of

values that are statistically random but have a definite starting point and are repeated over and over again. Equipment that generates such sequences will be labeled PN generators in this report. These PN sequences are used as spreading functions in the case of direct sequence spread spectrum techniques or as frequency determiners for frequency synthesizers in frequency hop spread spectrum.

Direct Sequence Spread Spectrum / Code Division Multiple Access

In direct-sequence spread spectrum (DSS), the message is multiplied by a wideband PN waveform before it is modulated. Multiplying the message by this PN waveform essentially masks the signal and spreads the spectrum of the signal (Carlson, Crilly, & Rutledge, 2002). It is similar to frequency modulation schemes but instead of the message causing the spectrum spreading, a PN sequence does the spreading.

Once the signal has been spread, it can be modulated. One way to accomplish this is to multiply the spread signal with a carrier. The general block diagram below shows how DSS is accomplished.

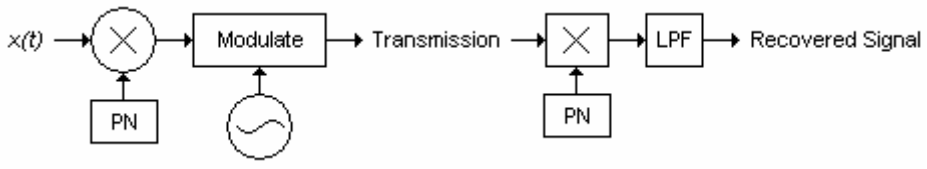

Figure 5. Simplified Block Diagram Showing DSS Operation

Code division multiple access (CDMA) is an extension of DSS in that, each channel is encoded using a unique PN sequence. In order to recover the message from a particular channel, the PN sequence corresponding to the channel must be known.

Resistance to Noise

During transmission, noise is picked up. If we let *z(t)* stand for additive noise introduced, $x_d(t)$ for the transmitted DSS signal, then the sum of the two signals make $y(t) = x_d(t) + z(t)$. At the receiver, the signal *y(t)* is multiplied by a PN sequence, *c(t)*. The resulting signal will be called *y'(t)*.

$$y'(t) = \left[x_d(t) + z(t) \right] c(t)$$

Since $x_d(t)$ is the transmitted DSS signal, multiplying it by c(t) will result in,

$$\begin{aligned} x_d(t) &= x(t) \times c(t) \\ x_d(t) \times c(t) &= x(t) \times c(t) \times c(t) \\ &= x(t) \times 1 \\ y'(t) &= x(t) + z(t) \times c(t) \end{aligned}$$

So, *y'(t)* becomes,

$$y'(t) = x(t) + z(t)c(t)$$

During the multiplication process, $z(t)c(t)$ effectively spreads the signal *z(t)*, which is the noise signal, but de-spreads $x_d(t)$ to recover *x(t)*, the original message signal. If lowpass filtering is performed, the out of band component of $z(t)c(t)$ is removed, which further reduces the effect of noise.

Frequency Hop Spread Spectrum

Frequency hop spread spectrum works in the following manner. Unlike DSS or CDMA, the message is FSK modulated before frequency hopping or spreading is applied. The FSK signal is then multiplied by the output of a frequency synthesizer that hops in frequency. The frequency synthesizer can hop to one of $Y = 2^k$ values, where k is equal to the number of bits coming in from the PN sequence generator at once.

There are two types of FHSS systems: slow hop and fast hop. In slow hop spread spectrum, one or more message symbols are transmitted per hop. However, in fast hop spread spectrum, several hops in frequency occur per message symbol. Because modules needed for fast hop was not purchased, only slow hop spread spectrum was investigated.

A benefit of using slow hop spread spectrum is that in order to recover the message, the same procedures used to recover the message from FSK signals can be utilized. A general block diagram implementation of FHSS is shown below.

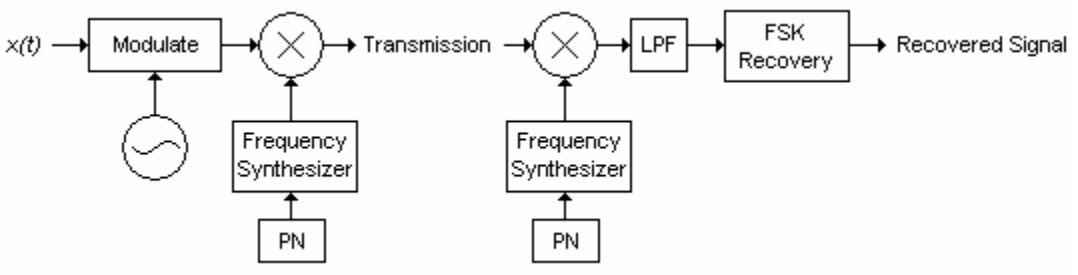

Figure 6. Simplified Slow Hop FHSS Block Diagram

Resistance to Noise and Interference

Similar to the way it was observed in CDMA, FHSS is highly resistant to noise and interference. In the process of recollecting spread signals, noise and interference is spread. This was also observed in CDMA.

In particular, FHSS systems elude interference well. Even if there are jammers that attempt to block a particular frequency or frequency range, the FHSS system can hop around it and recover the message from hops that are not blocked.

3. Project Overview

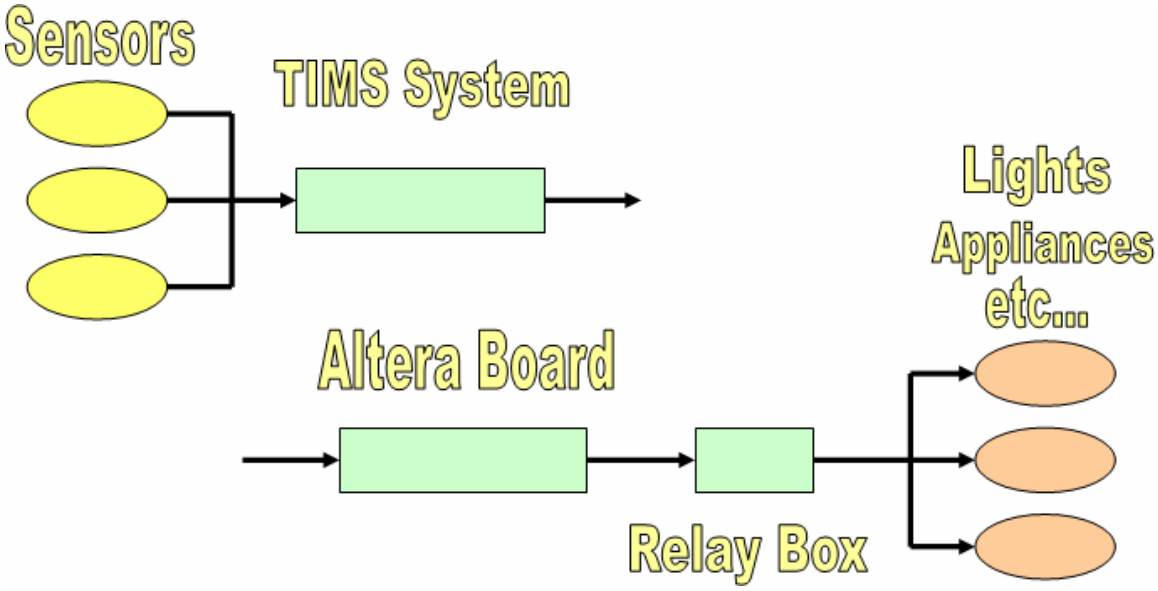

Figure 7. Project Overview and Implementation

The project can be divided in to several components as seen in Figure 7. At the front most end of the setup are the sensor circuits. The sensors, after detecting variables such as motion, force, or pressure, will send the information to the TIMS. Here, the signals will be modulated using various communication techniques and appropriate demodulation is performed. These signals are then sent to a Control Station where the signals are interpreted. An Altera Board will be used to act as the Control Station. Based on the program loaded on to the Altera Board, it will determine the appropriate outputs to send to the Relay Box. The Relay Box, which houses three solid state relays, will then switch on and off appliances.

Sensor Circuits

In this project, two types of sensors are used. One is an Infrared sensor used to detect motion and the other is a pressure sensor used to detect pressure or force on a surface.

Infrared

The components used to build the infrared sensor circuit were Lite-On Electronics LTE-4206 infrared emitters and Lite-On Electronics LTR-4206E infrared receivers. The emitter and receiver, along with other readily available components, were setup in the following manner.

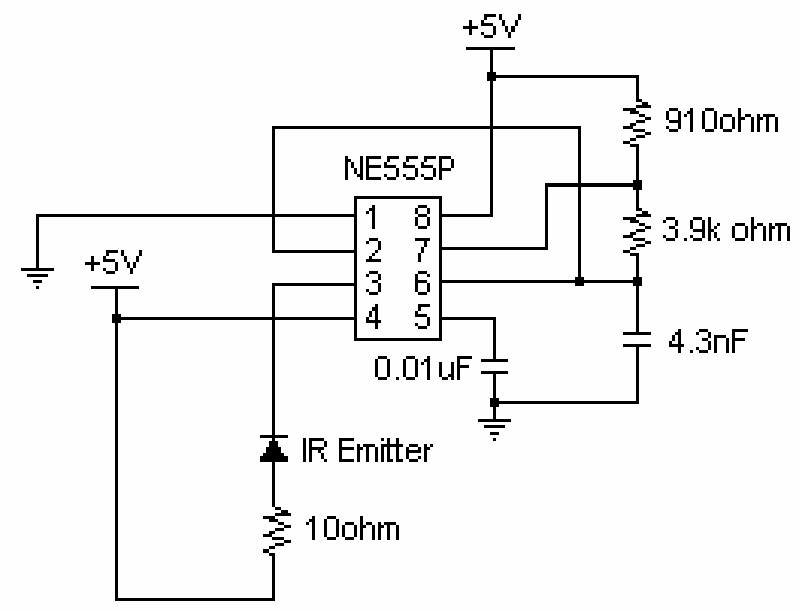

Figure 8. Infrared Emitter Circuit

The NE555P timer is used to drive current to the IR emitter. The timer, which is run in astable operation, is clocked at approximately 32 kHz and achieves a duty cycle of

44%. Having a high duty cycle allows the IR emitter to be driven with a current higher than it is rated for without damaging it.

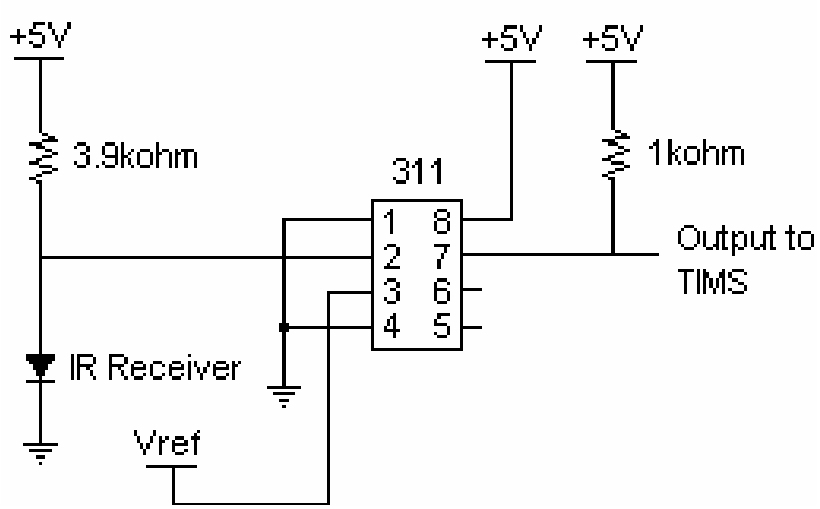

Figure 9. Infrared Receiver Circuit

The comparator in the receiver circuit essentially looks at the voltage appearing on pin 2 and compares it to the reference voltage coming in at pin 3. The input voltage will be low when line of sight is established between the emitter and the receiver. The more IR light hits the receiver, the greater the current flow through the receiver. On the other hand, if no light hits the receiver, then there is no current flow and the voltage at pin 2 appears to be 5 volts. If the input voltage is higher than the reference voltage, the output at pin 7 will be 5 volts. If the input voltage is lower than the reference voltage, the output is low or 0 volts. These outputs are then sent to the TIMS for transmission. The reference voltage was set at approximately 4 volts.

Pressure

The pressure sensor used in this project was the Thin-film FlexiForce A101 sensor. The sensor when connected to a circuit, acts as a resistor. With no load, the

sensor is approximately 20Mohms. With maximal load, the resistance decreases to approximately 20kohms. The sensor circuit built was based on the recommended circuit usage diagram from FlexiForce placed in the Appendix.

By changing the value of the feedback resistor R_F, it is possible to adjust the output voltage of the first stage, V_1. The output V_1 is in fact defined by the equation:

$$V_1 = -V_D \left(R_F / R_S \right)$$

Where R_F is the feedback resistance and R_S is the sensor resistance.

Then, connecting V_1 to a comparator and setting the proper reference voltage, we can obtain a binary output from the pressure sensor. The reference voltage can also be used to calibrate the sensitivity of the circuit. Once everything is properly calibrated, the data is sent to the TIMS system for transmission.

Wireless Transmission

Telecommunications Instructional Modeling System

As previously mentioned, wireless transmission was simulated using the Telecommunications Instructional Modeling System (TIMS) from Emona Instruments Pty. Limited. It is a communications trainer that allows the modeling of various communication methods. Modeling is done on the level of block diagrams and specific modules can be inserted in to the TIMS to build the desired system.

The TIMS is a very useful machine since all communication techniques can be examined on the block diagram level. This allows for quick and easy implementation of these techniques, while allowing the user to focus on the theoretical aspects of communication theory.

Control Station

Altera UP1 Programming Board

An Altera UP1 Programming Board was selected to act as the Control Station for this project. Altera Boards can be quickly and easily loaded with different programs to suit different situations. For this project, the Altera Max EPM7128SLC84-7 Chip was used over the Flex10K chip due to the non-volatile nature of the memory used in the EPM7128SLC84-7.

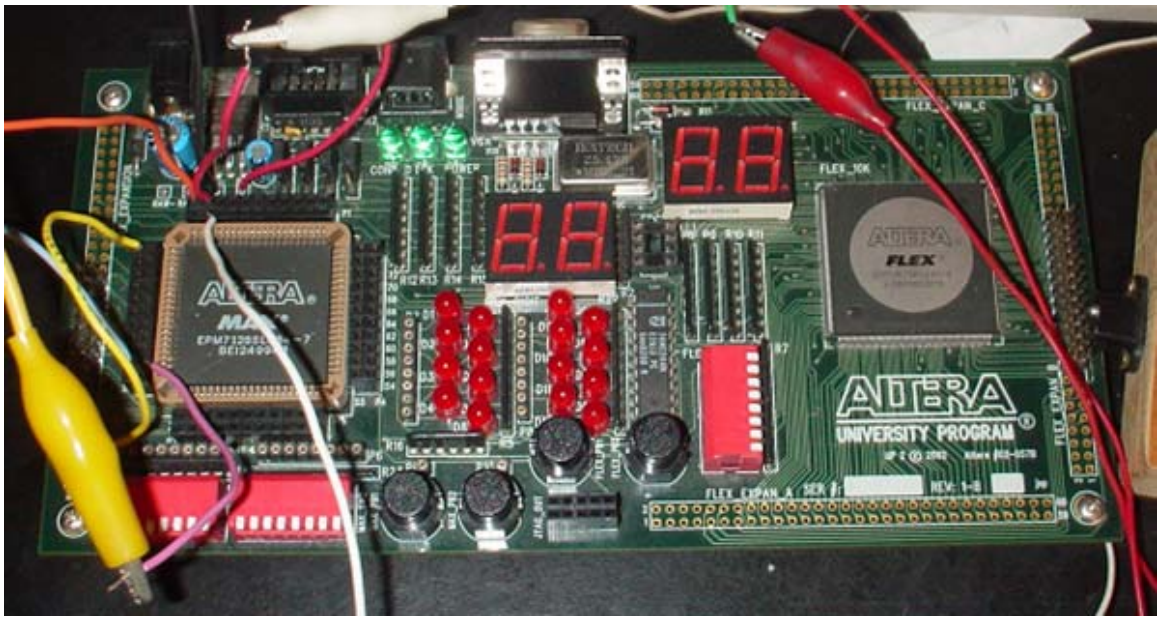

Figure 10. An Altera Board was chosen to act as the Control Station

Furthermore, the standard power supply and the onboard regulator were bypassed. Instead, power was drawn from a bread board to match ground and +5V with the rest of the setup. This was necessary for proper interfacing with other components of the project, primarily the TIMS and the relay box.

Control Programs

Several control programs were written for this project to meet different applications in which the sensor network can be used. The programs were programmed in a language called VHDL, which stands for VHSIC Hardware Description Language. The VHSIC stands for Very High Speed Integrated Circuit. VHDL is a language that describes the behavior of digital systems. Since the outputs arriving from the sensors are digital signals, the control program determines the course of action the control station is to take and outputs the respective signals to a relay box that controls the activation of particular devices.

Application: ControlV5.vhd

> *Two infra-red sensors monitor the entrance of the room. Upon entry, the room lights and fan are turned on and a local monitor is activated. The local monitor checks a pressure sensor to see if anybody has sat down at the desk. If it detects a presence, then a local appliance is activated.*

Structurally, the program was coded in the following manner. There are three state-machines operating simultaneously. The first state-machine, called the MainMonitor, monitors activity of the sensors at the entrance of the room and keeps track of movement in to and out of the room. A population counter is maintained and when no one is present in the room, all appliances are turned off.

A second state-machine, called the LocalMonitor, monitors local sensors within the room. In this program, a pressure sensor was placed at a chair and detected whether or not someone was sitting on the chair. However, the LocalMonitor will not go active unless the MainMonitor has detected entry and presence in the room.

Finally, the third state machine controls the outputs to the Relay Box and external displays based on flags triggered by the Main and LocalMonitors.

The entrance monitoring scheme discussed for the MainMonitor was inspired by the Head Counter (Virtual Door) setup used by Antenah Tesfaye and David Whitehead (2003). Modifications were made but the general idea remains the same. To implement the MainMonitor, a pair of infra-red sensors was used to detect movement in to and out of the room. Using a pair of sensors allowed for the detection of direction based on the order of sensor triggering. If the sensor located outside the room was triggered first, followed by the inner sensor, the direction of motion is in to the room. If the sensors were triggered the other way around, inner first then outer, the direction of motion is out of the room. This allows for the accurate tracking of entry and exit from the room, as well as the ability to keep track of the number of people present in the room.

When there is no one present in the room, as deemed by the MainMonitor, the LocalMonitor and OutputMonitor remain in the inactive state. If two way communications could be established in this project, this could allow for the turning off of power supplies or batteries to local sensor circuits, with the effect of conserving energy and cost. Due to the limit in the number of modules useable at once in the TIMS, this option could not be explored. However, the control program is highly customizable and updateable in the event that future projects would like to pursue this path.

All vhd files for this application have been placed in the Appendix.

MainMonitor

In order to completely capture activity at the entrance of the room, the MainMonitor was broken down in to 5 different states: Snoop, EntryDetect, Entry Count,

ExitDetect, and ExitCount. The state diagram representing the MainMonitor can be seen in Figure 11 below.

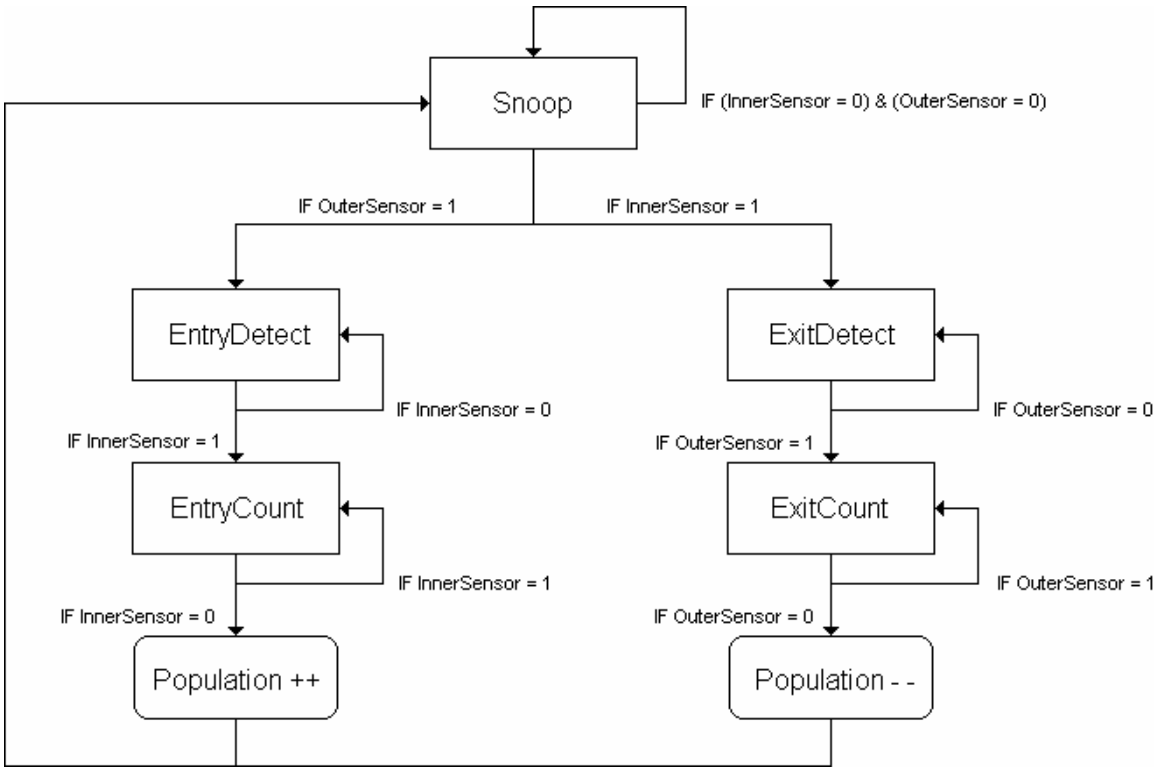

Figure 11. State Diagram for the MainMonitor

In the Snoop state, the MainMonitor examines the input lines coming in from the two entrance infra-red sensors. If neither is high, the MainMonitor remains in the Snoop state. When the outer sensor line is high, the MainMonitor goes to the EntryDetect state, where it waits for the inner sensor line to trigger. When this triggers, the MainMonitor then goes to the EntryCount state, where it waits for the inner sensor line to go low and then increments the population counter. Upon completion of this, the MainMonitor is returned to the Snoop state.

A similar process takes place when the inner sensor is triggered first. In this case, however, the MainMonitor moves from the Snoop state to the ExitDetect state, where it

waits for the outer sensor to trigger. When triggered, this will send the MainMonitor to the ExitCount state and wait for the outer sensor line to go low. Once this occurs, the population counter is decremented and the MainMonitor is returned to the Snoop state.

LocalMonitor

The LocalMonitor is structurally very different from the MainMonitor. In this scenario, the LocalMonitor monitors the activity of one pressure sensor. The LocalMonitor is made up of 4 states: Idle, Active, SetOut, and SetDelay.

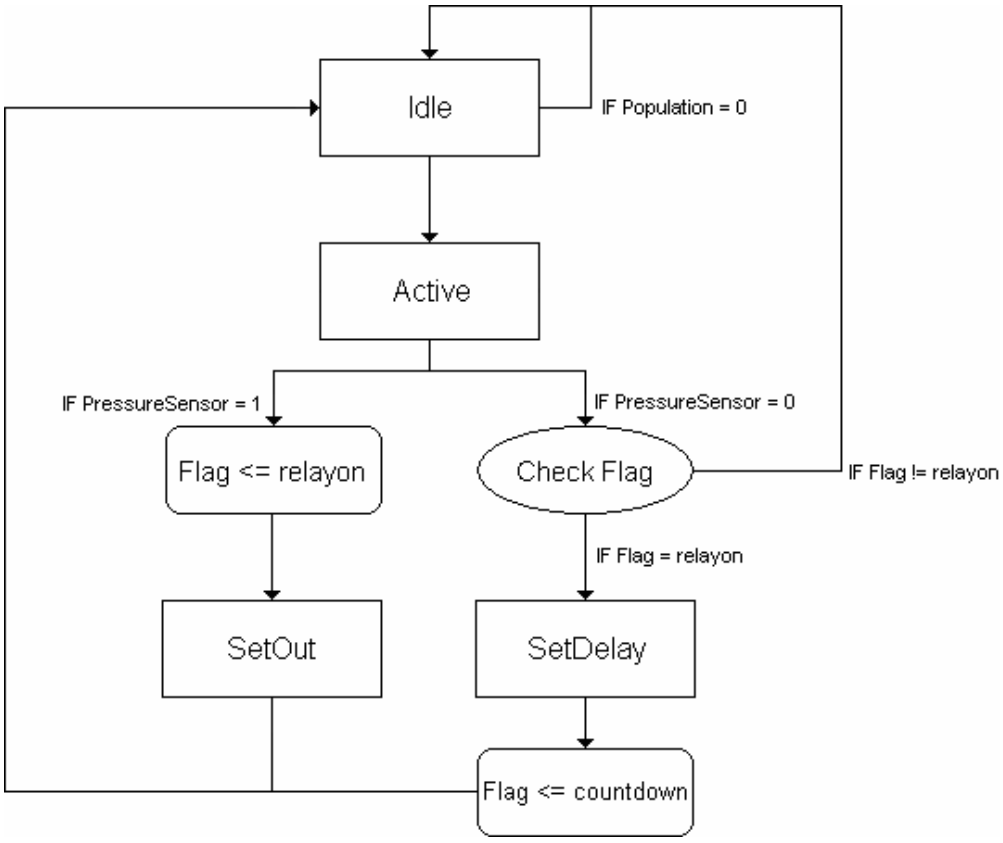

Figure 12. State Diagram for the LocalMonitor

The LocalMonitor remains in the Idle state as long as the population counter set by the MainMonitor remains zero. However, the moment the population is greater than zero, the LocalMonitor goes in to the Active state, where it will snoop the input line

coming from the pressure sensor circuit. If the line is high (i.e. if someone is sitting on the chair), the LocalMonitor enters the SetOut state. In this state, a flag will be flipped, indicating that the local appliance should be turned on.

The LocalMonitor then returns to the Active state and continues to monitor the input line. When the monitor detects that the line has gone low, it will enter the SetDelay state. In this state, a delay timer flag will be flipped, indicating that a delay timer should be activated. This delay timer is simply a short timer that delays the turning off of the appliance. For instance, if a person sitting at a desk reaches up for a book and leaves the chair on which the pressure sensor is mounted for a few seconds, instead of having the appliance turn off and then on once the person returns, a delay timer will prevent the system from performing unnecessary switching. This delay timer will reset once the person returns and will not turn activate until the person leaves the chair again. The actual countdown process of this timer is performed in the OutputMonitor.

OutputMonitor

The OutputMonitor is a state machine that controls the actual output signals being sent to the Relay Box based on the flags set by the Main and LocalMonitor. It is also responsible for displaying the number of people present in the room based on the population counter and handling timer requests from the LocalMonitor. Furthermore, the OutputMonitor will remain in the Idle state as long as the population is zero, as was the case with the LocalMonitor. In the Idle state, all outputs to the Relay Box are low and, hence, all appliances will be off.

Relay Box

A relay box, housing three solid state relays, was used to control appliances used in the project. Standard solid state relays from Teledyne, Kodak, and Potter and Brumfield were used. A solid state relay has four connection points as shown below.

Figure 13. Teledyne Solid State Relay

Points 1 and 2 are used to complete the connection to a wall outlet. Points 3 and 4 make the control line. When a high signal is sent across the control line, a connection between points 1 and 2 is established. When the control signal is low, there is no connection between points 1 and 2.

Appliances and the Altera board outputs were connected to the relays in the following manner.

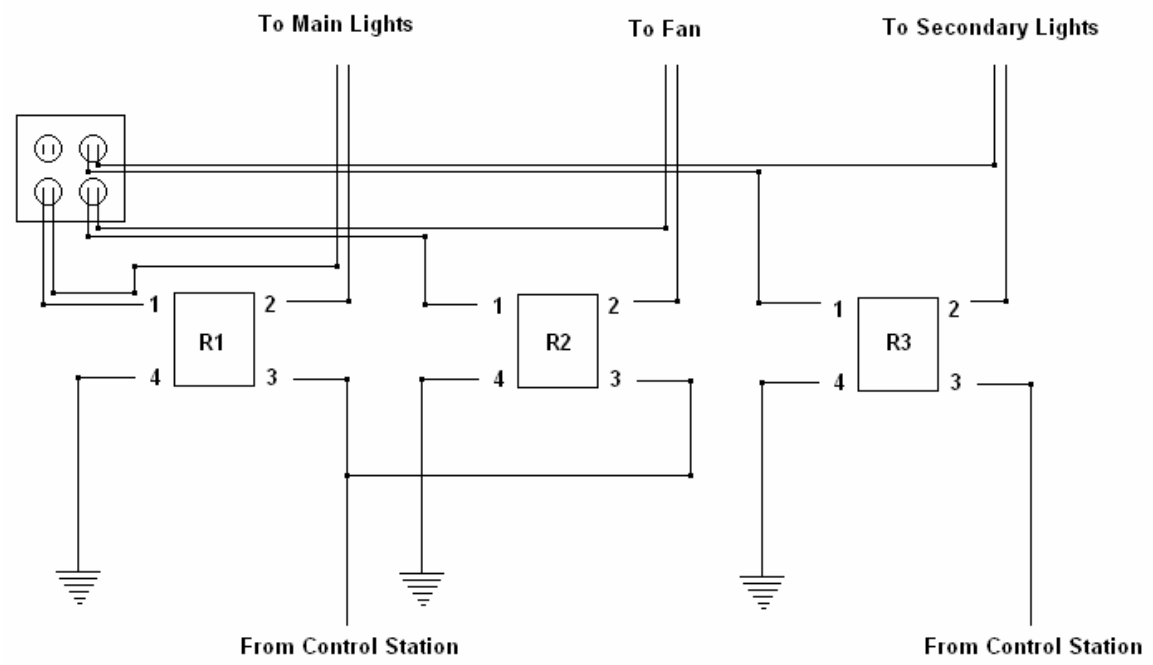

Figure 14. Simplified Connection Diagram for the Relay Box

4. Results

Sensor Circuit

The infrared sensors were able to attain a range of approximately 3 feet. Furthermore, since the sensors need to line up almost perfectly straight, interference from a second set of sensors was not observed despite being only a few inches apart. However, the sensors were found to be sensitive to external light sources if pointed at them. Perhaps as an improvement, low pass filters could be added to remove environmental noise from registering at the receiver.

Pressure sensors registered worked well with the given circuit. Varying the reference voltage allowed for the adjustment in the sensitivity of the sensor. The same could be performed by changing the feedback resistors. For testing purposes, instead of mounting the sensors on chairs, they were mounted on wooden blocks to be pressed down on with one's finger. After calibration, a firm press on the sensors would register correctly.

Wireless Transmission

Amplitude Shift Keying

Amplitude shift keying was performed on the TIMS by using the following modules: Dual Analog Switch and Audio Oscillator. The Audio Oscillator was connected to an 8 kHz TTL signal which simply produces a sinusoidal of 8 kHz. This was connected to one of the inputs on the Dual Analog Switch. The other input on the switch was connected to a set of infrared sensors. When the signal was from the sensor

was high, the output was an 8 kHz sinusoidal. When the signal was a low, the output was flat.

Another method to generate ASK signals is to simply multiply the input data and the carrier. If the input is at 0V, then the resulting ASK signal is 0V. If the input is 5V, the resulting signal will have a carrier frequency.

However, using ASK results in a rather wide bandwidth. This arises from the sharp discontinuities in the signal. Bandlimiting can be performed by passing the signal through a band pass filter after ASK has been performed or by passing the sensor output through a low pass filter before ASK has been performed.

To demodulate the signal, envelop detection can be performed. The transmitted ASK signal can be passed through a rectifier and then through a low pass filter. This smoothes out the higher frequency components and leaves the envelope of the transmitted ASK signal. Passing this signal through a comparator gives a sharper output.

Phase Shift Keying

BPSK can be performed by multiplying the carrier with a bipolar signal. Since the output of the sensor circuits are 0V or 5V, a DC voltage of -2.5V was added to the sensor outputs, yielding a signal of -2.5V or 2.5V. This signal was then multiplied with the carrier signal to produce a BPSK signal. Once again, discontinuities produce unnecessarily wide bandwidth and needs to be bandlimited. As was the case with ASK, there are two areas where bandlimiting can be introduced. It could be done by applying a lowpass filter before the sensor output is modulated or a bandpass filter could be applied after carrier modulation.

Demodulation of BPSK was performed by multiplying the BPSK signal with the carrier that was used to modulate it. The carrier was passed through a phase shifter so adjustment in phase could be made to obtain the strongest recovered signal. After the multiplication process, the signal was passed through a low pass filter. Since the resulting signal is slightly noisy due to the bandlimiting and demodulation process, a comparator can be used to obtain a sharp output, as was the case with ASK.

QPSK can be obtained by performing the same process as BPSK except using two carriers that are 90° off in phase. For example, one sensor output can be modulated using a sinusoidal carrier at 100 kHz and another sensor output can be modulated using a cosine carrier at 100 kHz. These signals are then added to produce a QPSK signal.

Once this signal is transmitted, recovery can be performed. Once again, the recovery process is almost identical to the BPSK case. The only difference is that the transmitted signal is multiplied by the respective carrier used to modulate the message to recovery the message. The multiplied output is passed through a low pass filter and then through a comparator to get clean outputs.

Frequency Shift Keying

In order to implement FSK on the TIMS, the following block diagram was followed.

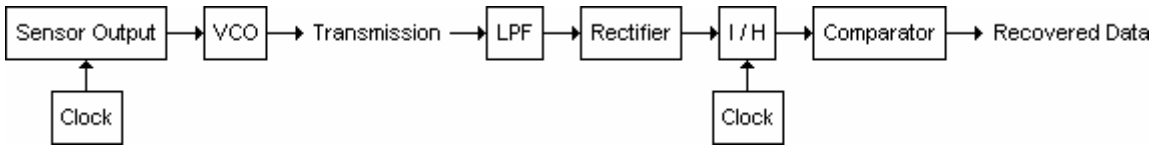

Figure 15. Implementation of FSK

First, the data input from the sensors were sampled and held. This was necessary in order for the integrate and hold operation in the recovery process to work. Then, using

a voltage controlled oscillator, the input data was modulated on to a signal with two carriers. A 5V input was associated with 4.7 kHz and a 0V input was associated with a 9.8 kHz carrier. This can be seen in signal 2 of the oscilloscope output shown in Figure 16a, along with the original input data, signal 1. Here, however, the input data was taken from a PN generator for demonstration purposes.

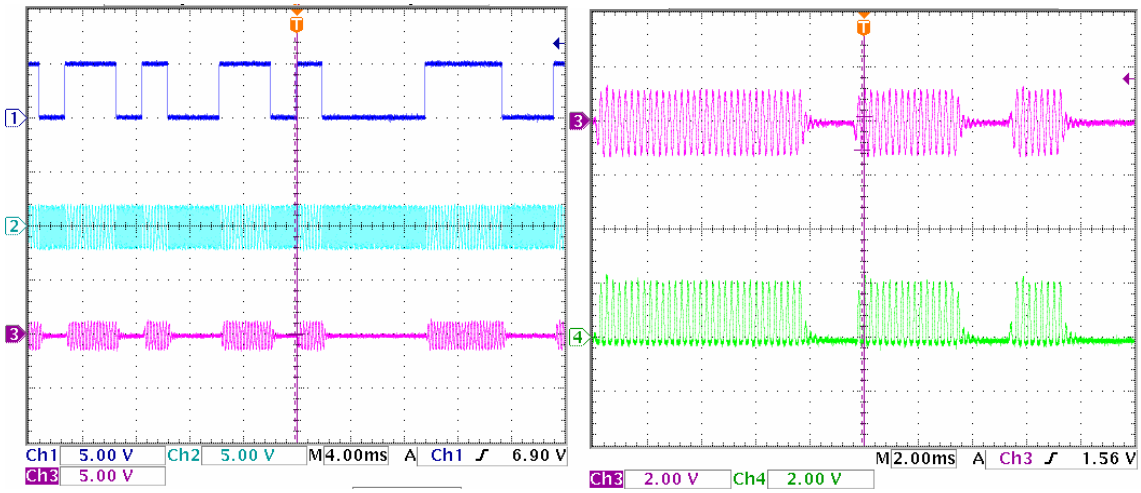

Figure 16. (a) Oscilloscope Output of the Input Data, Modulated Signal, and Low Pass Filtered Signal. (b) Oscilloscope Output Showing Low Pass Filtered Signal and Rectified Signal

Signal 2 from Figure 16a is transmitted and at the receiving end, it is low pass filtered. A cutoff frequency of approximately 5.2 kHz is used. This value was obtained empirically by looking for a cutoff frequency that attenuated the 9.8 kHz component and maintained the 4.7 kHz component. After lowpass filtering is performed, the signal is rectified as shown in Figure 16b.

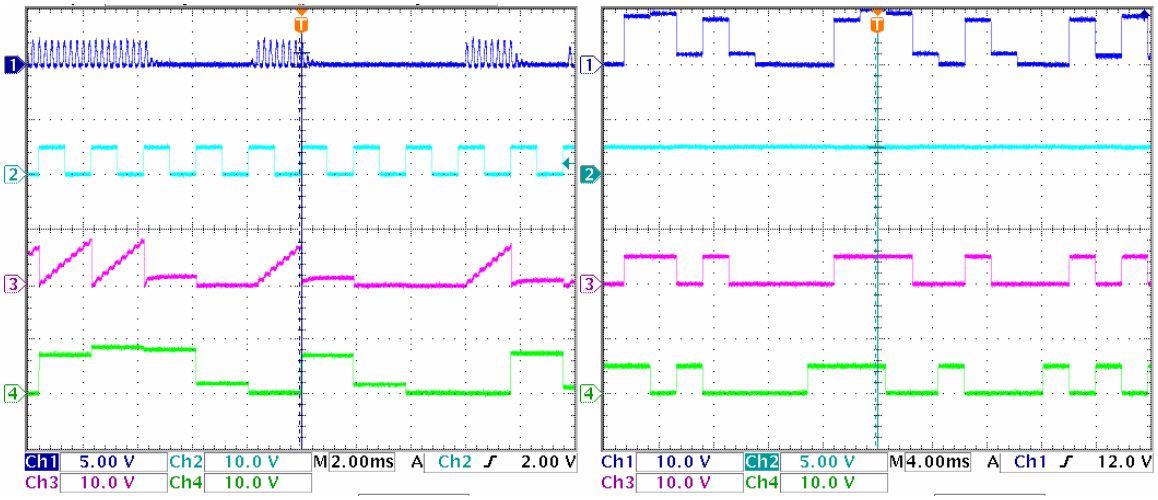

Figure 17. (a) Oscilloscope Outputs of Rectified Signal, Data Clock, Integrate and Dump Operation, Integrate and Hold Operation. (b) Oscilloscope Output of Integrate and Hold Operation, Reference Voltage for the Comparator, Recovered Data, Original Input Data.

The rectified signal is then integrated and held using the Sample and Hold module. A clock is needed for this operation in order to specify when the next integrate and hold should be performed. The clock used to sample and hold the data input signal was used and is shown in Figure 17a. The resulting intgrate and hold operation performed on the rectified signal is also shown on Figure 17a as signal 4.

Once the integrate and hold is performed, it is possible to see that by setting a reference voltage and passing the signal through a comparator, we can obtain a binary output. This process is shown in Figure 17b. When compared against the original input data, it matches except with a slight delay due to transmission and primarily because of the integrate operation.

While the oscilloscope outputs are based on random PN sequences generated, the procedure was tested with actual sensor outputs and communication between the sensors and control station was established.

Direct Sequence Spread Spectrum / Code Division Multiple Access

The following block diagram was used to implement CDMA. Both single channel DSS and dual channel CDMA was performed. Here, 2-channel CDMA will be examined since it encompasses single channel CDMA or DSS. Data for the two channels were obtained from slowly clocked, approximately 500 Hz, PN generators for this demonstration. However, for testing of the project, data from the sensors were used.

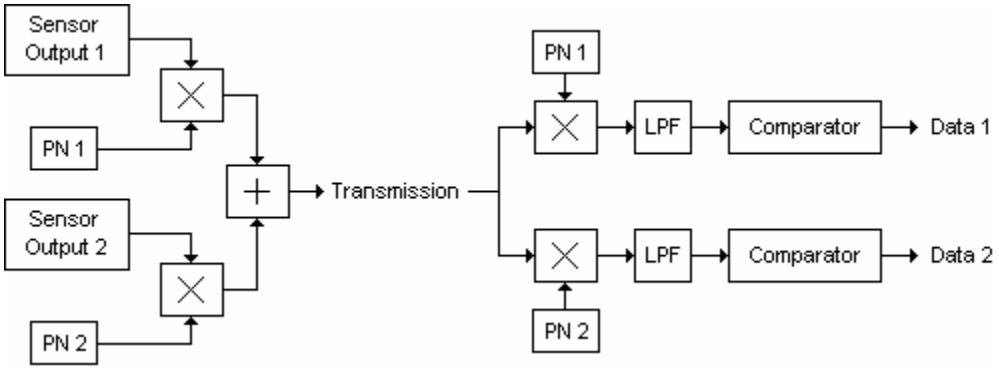

Figure 18. Block diagram of 2 channel CDMA

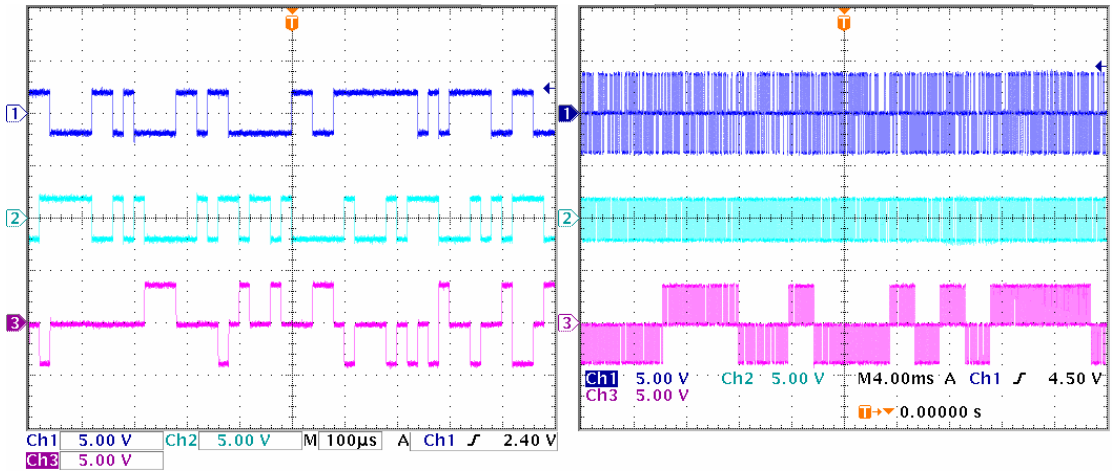

Figure 19. (a) Data 1 spread using PN sequence 1, data 2 spread using PN sequence 2, addition of channel 1 and channel 2. (b) Transmitted signal, PN sequence 1, multiplied result of transmitted signal and PN sequence 1

First, data from channel 1 and 2 obtained from the slow clocked PN generators are spread using two different PN sequence clocked at a much higher rate of 50 kHz. The

two signals are added and transmitted, as shown in signal 3 of Figure 19a. Signal 1 in Figure 19b shows the same signal except the x-axis has been compressed.

To recover the data from channel 1, the transmitted signal is multiplied by the highly clocked spreading function, PN sequence 1, shown as signal 2 on Figure 19b. The resulting signal is shown as signal 3 on Figure 19b. The resulting signal remains because of the mathematical properties, 1 x 1 = 1 and -1 x -1 = 1. The PN sequence essentially cancels itself out of the signal. Thus, the remaining components of the signal are now the original message of channel 1 clocked at 500 Hz and the spread frequency component of channel 2. An intuitive step would be to low pass filter the signal to remove as much of the channel 2 component that is spread in frequency.

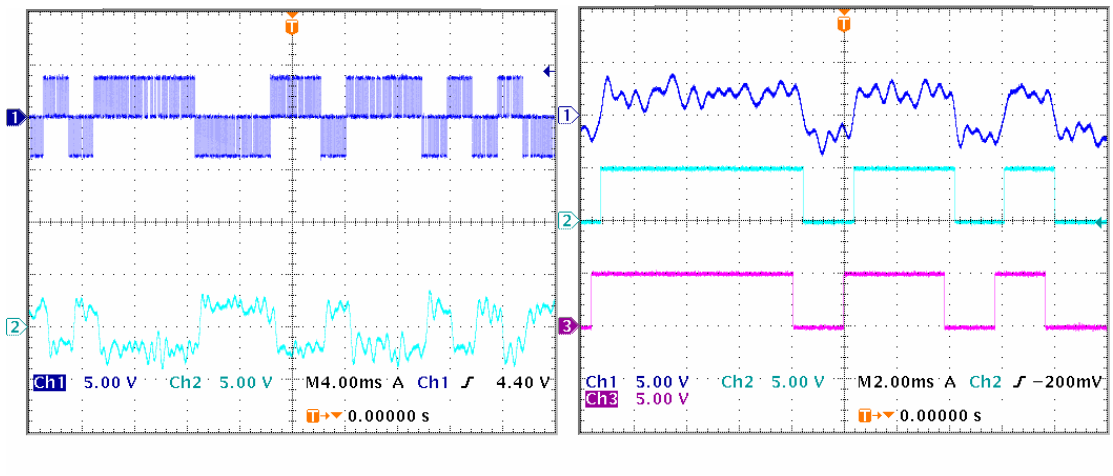

Figure 20. (a) Multiplied result of transmitted signal and PN sequence 1, low pass filter of multiplied result. (b) Low pass filtered signal, low pass filtered signal passed through a comparator, original message in channel 1

Performing a low pass filter yields signal 2 shown in Figure 20a. It is possible to observe the outline of a bipolar signal. Passing this signal through a comparator, we obtain signal 2 in Figure 20b. This matches very closely with signal 3 of Figure 20b, which is the original data sequence of channel 1.

While the signal was properly obtained, it brings to light an inherent problem with CDMA systems. When more channels are added, there is a greater chance of error due to the fact that all channels are simply being spread in frequency. When information from a single channel is wanted, small components of other channels are also being obtained. If enough channels are being used, it could corrupt data transmission by adding too much noise or excess signals. It is important, therefore, to minimize the cross-correlation between spreading codes for minimal interference between CDMA channels.

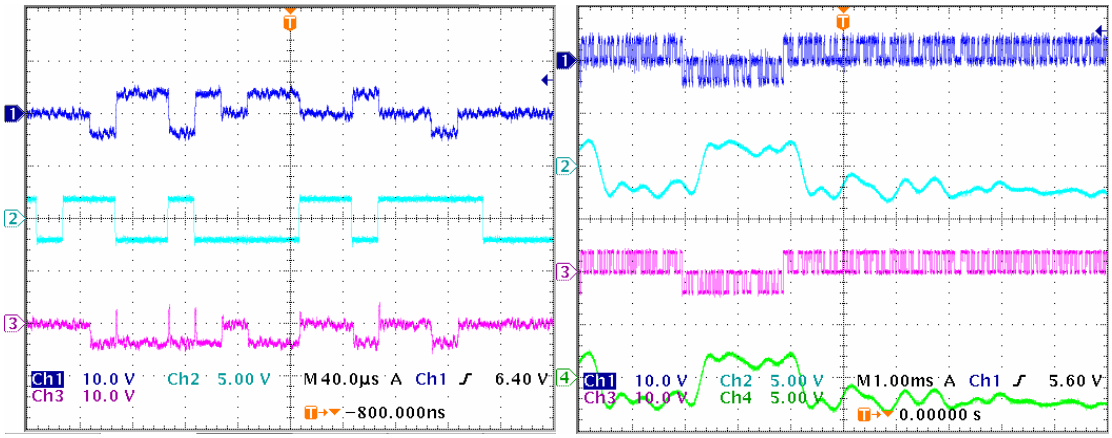

Figure 21. (a) 22dB noise added during transmission, PN sequence 1, multiplied result of transmitted signal and PN sequence 1. (b) multiplied result of transmitted signal with noise and PN sequence 1, low pass filter of multiplied result (with noise), multiplied result of transmitted signal with no noise and PN sequence 1, low pass filter of multiplied result (no noise)

The effect of noise on CDMA signal transmission is considered next. Figure 21a, shows 22dB noise added to the CDMA signal during transmission. When multiplied by its respective PN sequence, the waveform of signal 3 in Figure 21a is attained. Figure 21b shows the described waveform at signal 1, except on a different time scale. When low pass filtered, the resulting waveform of signal 2 is obtained. Here a comparison can be made with the waveform that would result had there not been any noise during transmission, which is shown as signals 3 and 4 in Figure 21b.

It can be observed that noise does not affect the CDMA signal. The resulting demodulated signal is almost identical to the case had there not been any noise. Passing the signal through a comparator will recover the proper signal. This can be attributed to the fact that CDMA spreads noise over its entire frequency band. Thus, the contribution of noise becomes smaller.

Frequency Hop Spread Spectrum

To implement FHSS, the following block diagram was assembled using the TIMS.

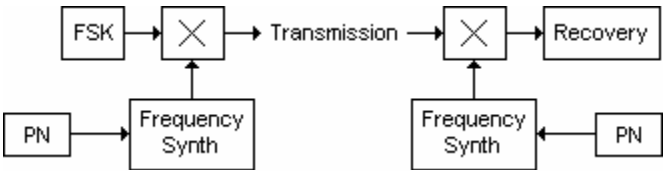

Figure 22. Implementation of FHSS on the TIMS

An FSK signal was generated, as discussed in the FSK section, and then multiplied by a varying modulating carrier. This signal was transmitted and then recovered by multiplying the received signal with another frequency synthesizer output that was controlled by the same PN sequence used to control the frequency synthesizer at the transmitting end. The clock for the PN generators in the transmitting and receiving ends were matched by stealing the clock. However, in practice, it is not a trivial process and simply stealing of the clock is not possible. It must be detected from the transmitted signal using feedback loops. The process is beyond the scope of this project and was not explored further.

To perform 2-channel FHSS, each frequency synthesizer was operated by different PN generators using different PN sequences. Since only 2 frequency synthesizers were available for use, each frequency synthesizer was used in both the

generation and recovery process. In this section, only one channel is discussed but in testing, 2-channel FHSS was performed.

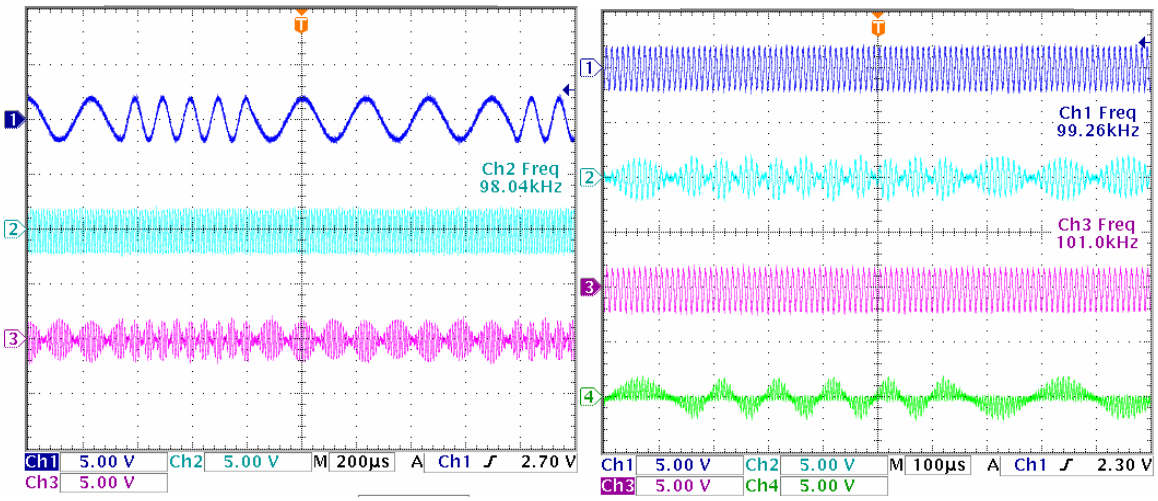

Figure 23. (a) FSK signal, carrier frequency at 100 kHz, Multiplication of FSK and 100 kHz carrier. (b) Frequency synthesizer output at transmitting side, transmitted signal, frequency synthesizer output on receiving side, multiplied result of transmitted signal and frequency synthesizer output on receiving side when synthesizers matching.

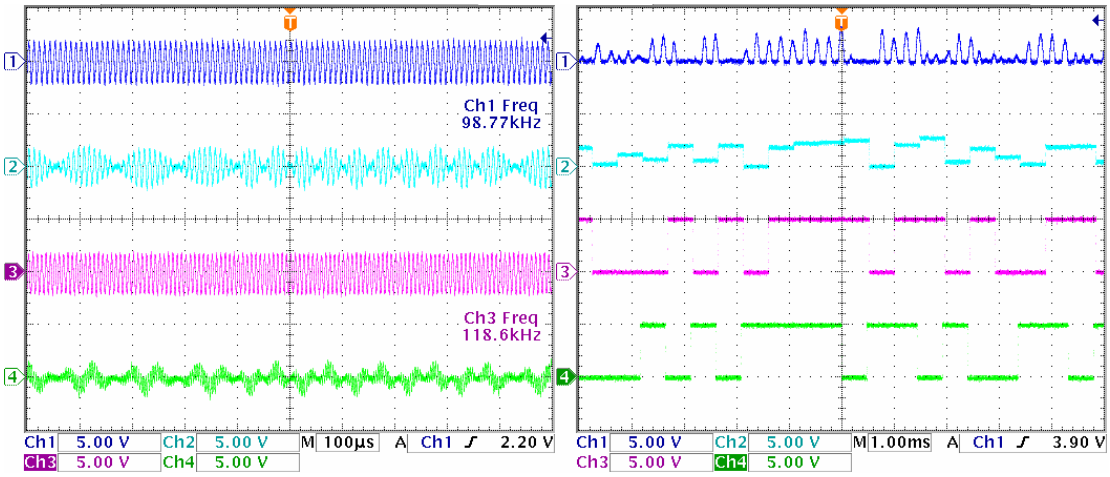

Figure 24. (a) Frequency synthsizer output at transmitting side, transmitted signal, frequency synthesizer output on receiving side, multiplied result of transmitted signal and frequency synthsizer output on receiving side when not synthesizers not matching.

Figure 23a shows the generation of a FHSS at a particular hop frequency, in this case 100 kHz. Typically, the carrier will hop once per bit. However, for demonstration

purposes, the hop carrier was manually controlled. When testing with sensors, the frequency synthesizers were driven by random PN sequences and slow hop spread spectrum was performed.

To recover the message, the transmitted signal is first multiplied by a signal with the same hop frequency in which the message was modulated using. The result is shown in signal 4 of Figure 23b. It can be observed that the resulting signal resembles the modulated data sequence shown as signal 2 of Figure 23b. The resulting signal is essentially a FSK signal. The same procedure used to obtain the message in the case of the FSK can be applied here. This process is shown in Figure 24b. The signal is low pass filtered, rectified, integrate and held, and passed through a comparator. As shown in Figure 24b, the recovered data, signal 3, matches the input data, signal 4.

In the case that the frequency synthesizers do not output a signal of the same frequency, as shown in Figure 24a, the resulting signal, signal 4, is not decipherable. It holds no resemblance to the modulated FSK or signal 2 in Figure 24a. When the rest of the recovery process is attempted, the data does not match the input data. This brings up the secure nature of FHSS systems. If the frequency generators are not matched, recovery of the signal is not possible. In order for the frequency generators to be matched, they have to be driven by the same PN code. Therefore, both the transmitting and receiving party must know the code in order to establish communication. As mentioned earlier, the PN codes also have to be aligned since they have a distinct start and end point. This process is not simple and hard to achieve in practice.

To observe the case in which the phase of the PN sequences are slightly mismatched, the frequency synthesizer on the receiver side was delayed. With minimal

delay, the resulting integrate and hold operation produces results shown in Figure 25a. With moderate delay, the resulting operation is shown in Figure 25b. When there is greater mismatch in the phase of the PN sequence, the recovered amplitude signal level decreases. If more delay is added, it will eventually reach a point where the signal will not be able to be recovered.

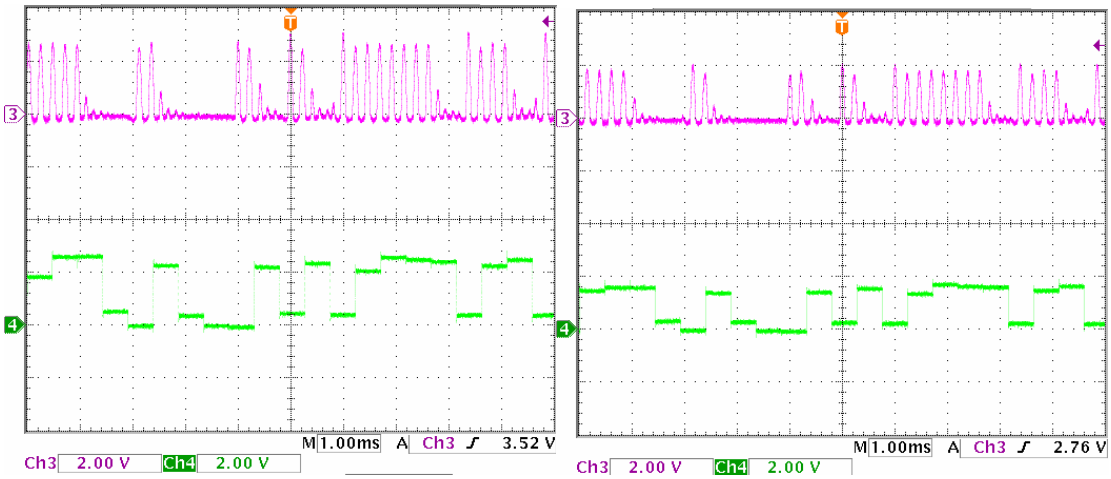

Figure 25. (a) Rectified signal, sample and held signal (minimal delay). (b) Rectified signal, sampled and held signal (moderate delay)

However, this information can also be used to align the mismatch in phase at the receiving end. Since recovered signal amplitude is greater with less phase delay, a feedback loop could be implemented to take advantage of this fact. A phase modifier could be implemented in the frequency synthesizer at the receiving end to carefully adjust the phase until the greatest signal amplitude is obtained and hence, signify a match in phase of the two synthesizers. Since this is beyond the scope of this project, it will be left as a potential path for future students to pursue.

Next, the effect of noise in FHSS systems will be explored. In the transmission stage, a 22dB random noise signal was added. The resulting transmitted signal can be observed in signal 1 of Figure 26a. Although the signal looks completely random, FHSS

signals are capable of recovering the message. It is important to mention here that, this is not a complete analysis of the effect on noise on FHSS systems since the hop carrier was manually adjusted. Since noise typically occupies a wide frequency range, and as long as the frequency synthesizer hops in the same range as the noise, simulating one frequency should provide similar results when hopping is performed.

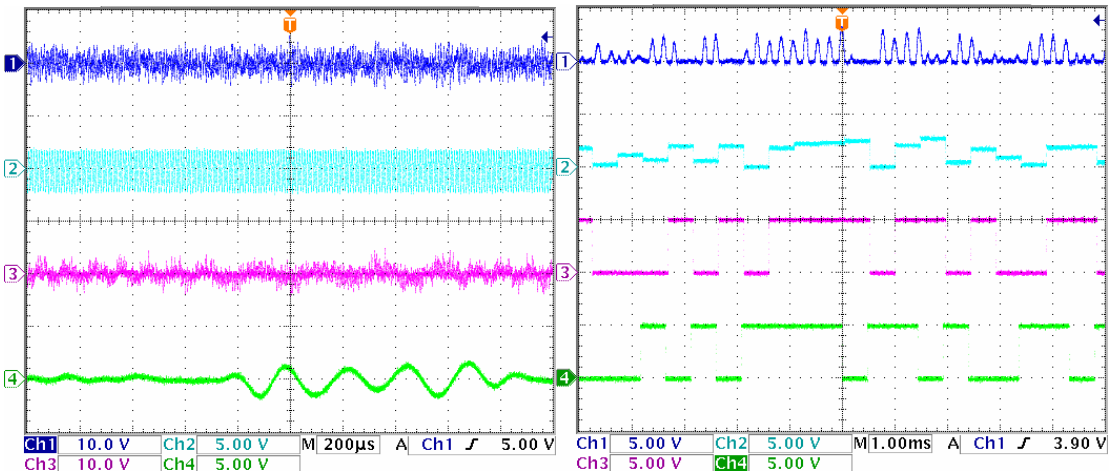

Figure 26. (a) Transmitted signal with noise, 100 kHz carrier, multiplied result of signal 1 and signal 2, lowpass filtered result of signal 3. (b) rectified signal, integrate and held, recovered message by passing through a comparator, original message.

After performing the recovery process, signal 3 on Figure 26b is obtained. This matches well with the original message, shown as signal 4 on the same figure. However, looking at the integrate and hold operation, signal 2, it is clear that noise does make the signal less clear. The reference level for the comparator has to be more finely adjusted in order to obtain the proper output. Therefore, it seems FHSS is not as noise resistant as CDMA. However, it should be noted that FHSS handles noise better than the digital baseband transmission techniques. This was the fullest extent to which noise analysis was performed due to time constraints.

Although the effects of jamming were not simulated, some conclusions can be made from the experiments performed for FHSS. Since jamming typically occupies one

frequency or a thin range of frequencies, it should have reduced effect on FHSS than on more conventional digital bandpass transmission systems. Still, slow hop spread spectrum will be more susceptible to jamming than fast hop spread spectrum. This is because slow hopping transmits one or more symbols per frequency hop. This means, if the frequency matches the jamming frequency, then the symbols will have to be retransmitted. However, in fast hopping, the synthesizer hops more than once per measure of information it wants to transmit. Therefore, the effects of jamming can be smoothed out since other frequencies will not be contaminated.

Simulation of Program Files

Now that various methods of data transmission have been attempted, the Control Station was tested to ensure it operated properly. Below are simulation results for ControlV5.vhd, which use two infrared sensors and one pressure sensor, as discussed in Section 3.

MainMonitor Operation

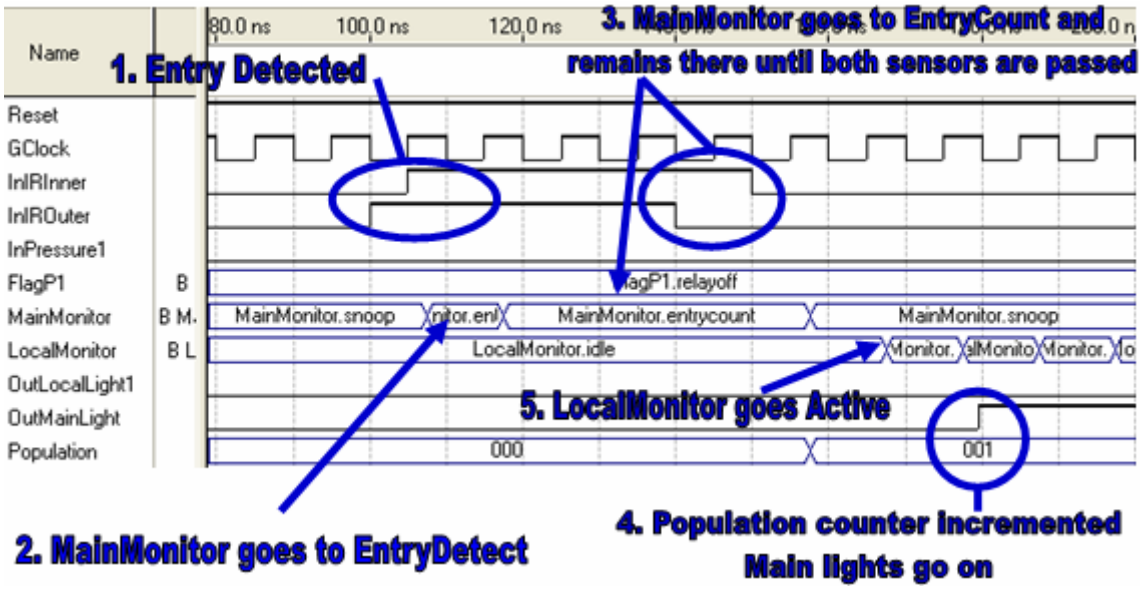

Figure 27. Simulation of MainMonitor Entry Detection

As can be seen, when the outer sensor is triggered first, MainMonitor goes to the state, EntryDetect. When the inner sensor is triggered, MainMonitor goes to EntryCount and remains there until both sensors have been passed. Once both sensors are passed, MainMonitor returns to the Snoop state and the population counter is incremented. At this time, the output signal for the main lights goes high. Furthermore, LocalMonitor remains idle and turns on when the population is incremented from 0 to 1.

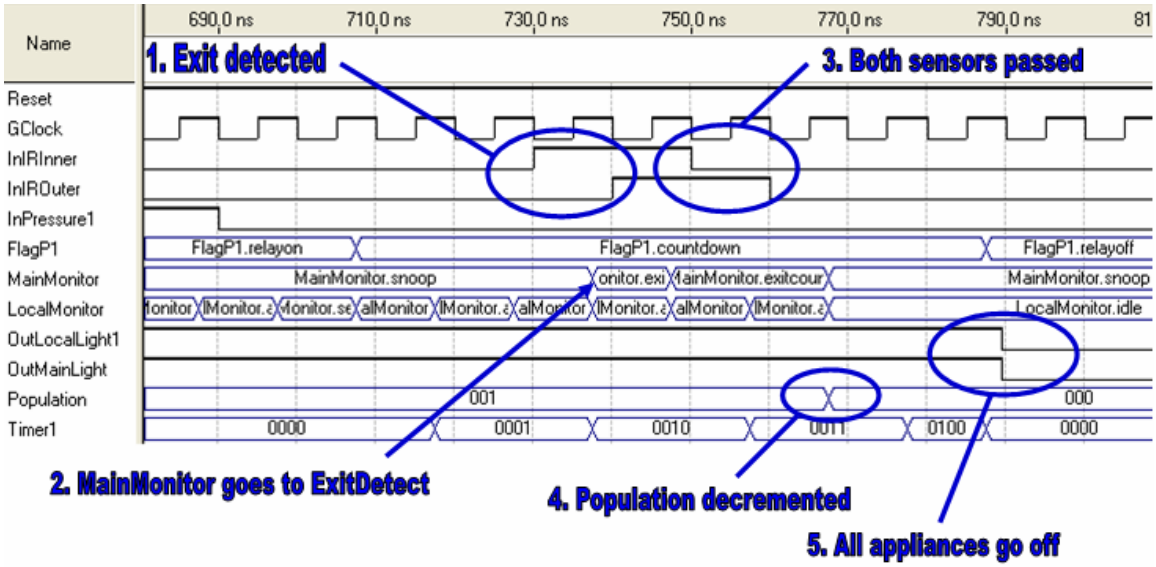

Figure 28. Simulation of MainMonitor Exit Detection

The above simulation represents the case when the last person in a room leaves. First, exit is detected when the inner IR sensor is triggered. This sends MainMonitor in to the ExitDetect state. Once the outer sensor is triggered, MainMonitor goes to ExitCount and remains in ExitCount until both sensors have been completely passed. When this occurs, MainMonitor goes to the Snoop state the population counter is decremented. Since the population is now zero, all appliances are turned off and the LocalMonitor is returned to the Idle state.

LocalMonitor Operation

Next, operation of the LocalMonitor will be examined.

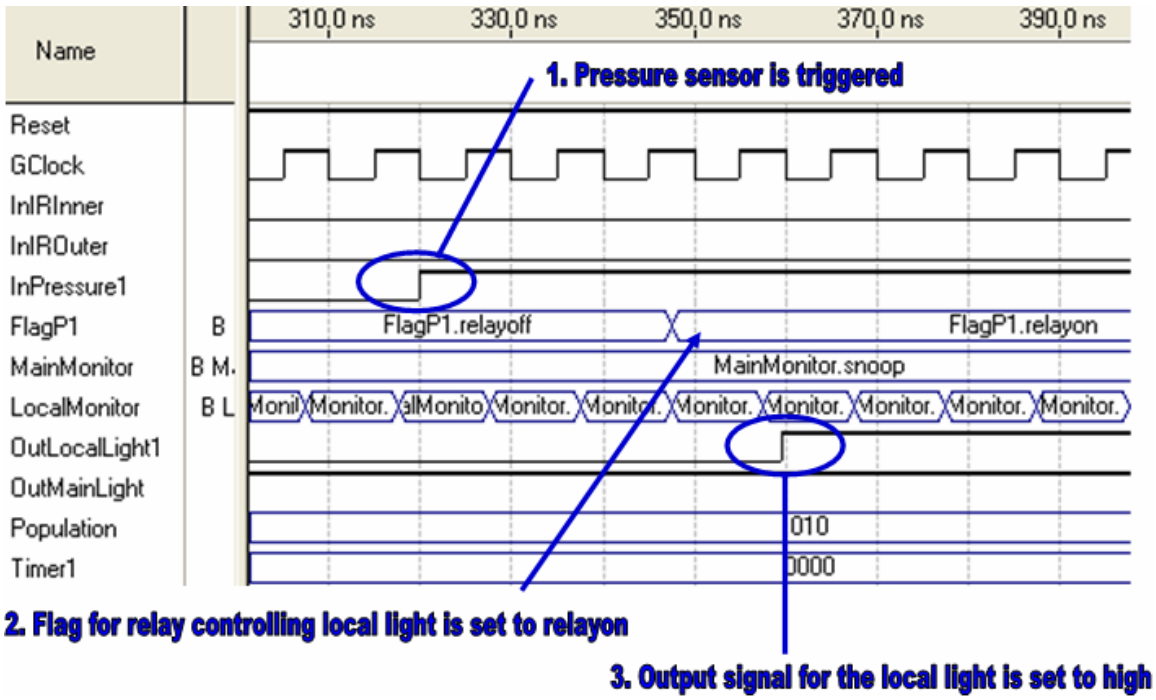

Figure 29. Simulation of LocalMonitor Pressure Sensor Detection

At point 1, activity at the pressure sensor has been detected. This flips a flag controlling the local light to relayon, as can be seen in point 2. Then, at point 3, the OutputMonitor sets the output for the relay to high.

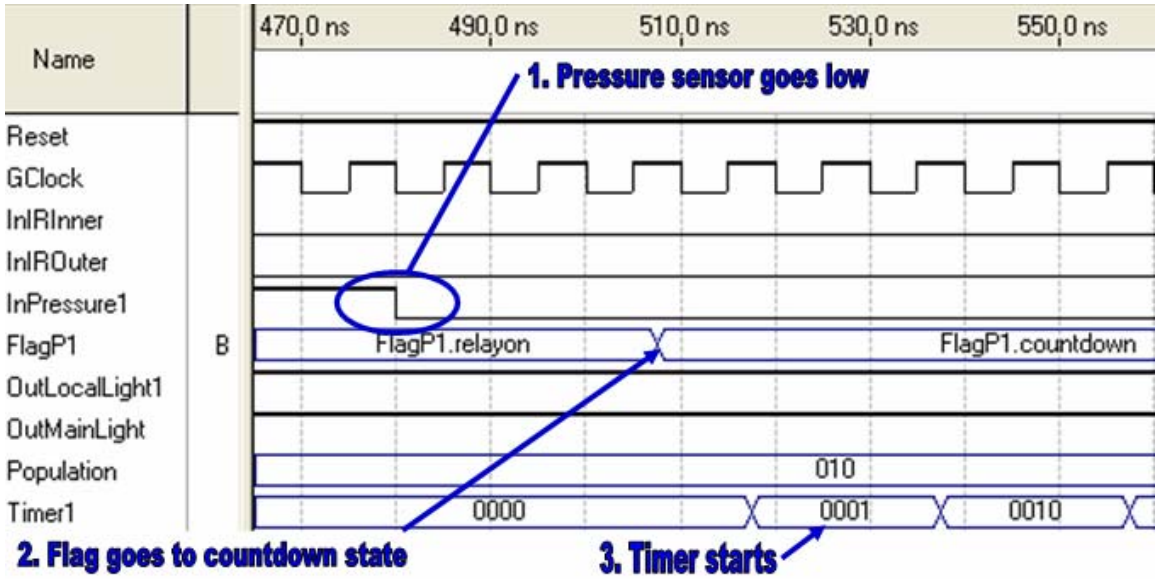

Figure 30. Simulation of Timer Feature

Here, the pressure sensor is observed to go low. When this happens, the flag for the local lights go in to the countdown mode. The timer is then started and begins counting up. During this time, the output for the local light remains high and the lights stay on.

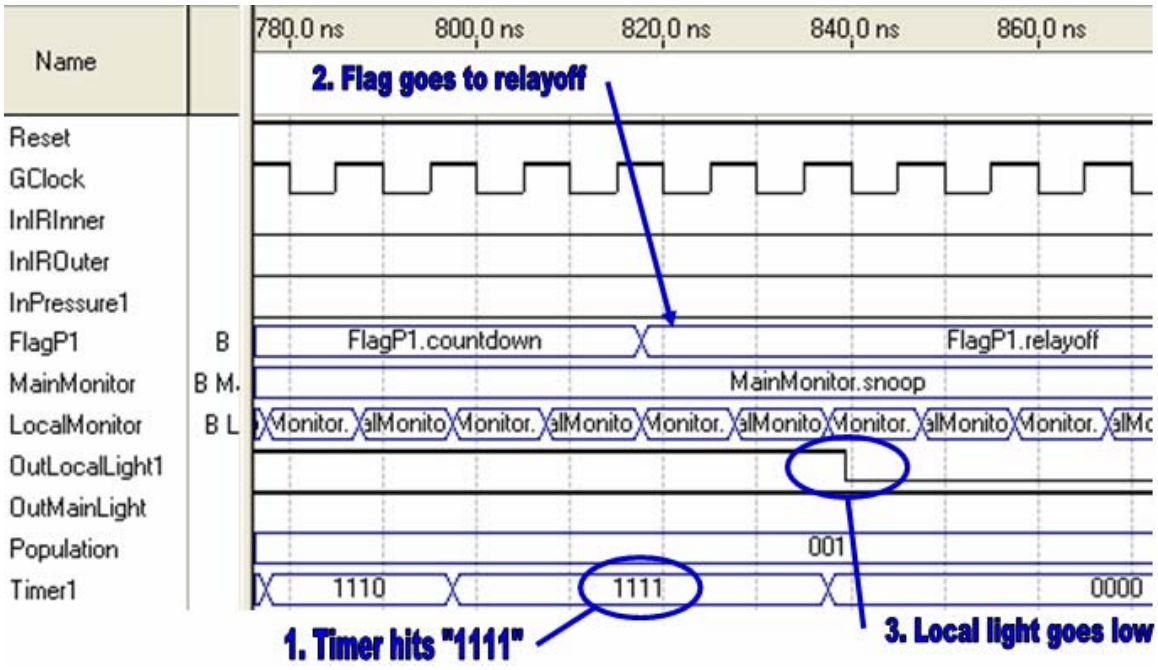

Figure 31. Simulation of Timer Termination

Once the timer hits “1111”, the flag for the local light is sent to the relayoff state. This is properly interpreted by the OutputMonitor and the output for the local light goes low. The simulations show proper operation of both the MainMonitor and LocalMonitor. Since all outputs are properly set, the OutputMonitor also works as expected.

5. Summary and Conclusions

In this project, the sensor circuits used in Tesfaye and Whitehead (2003) were modified and implemented to operate in a wireless sensor network.

Using the TIMS, various modes of communication were investigated. For purposes of sensor networks where noise and interference is a concern, spread spectrum techniques are preferable means of transmitting information from the sensors to the control station over more conventional digital bandpass techniques. Now that communication techniques have been investigated, antennas can be attached to perform actual wireless transmission.

The control station was successfully programmed and interfaced with the TIMS as well as the relay box. Application specific program was written and a delayed off feature was implemented for the local lights. The control station was capable of interpreting information coming from sensors and act accordingly by sending proper signals to the relay box. Solid state relays worked without a problem and successfully switched on and off appliances.

However, on certain occasions, external noise becomes a factor for the infrared sensors. Furthermore, the control system will interpret the passing of arms and the body of a person as separate people. Currently, it is only capable of recognizing one person passing through at a time. There were no issues with the pressure sensors.

6. Future Work

Below is a listing of potential future projects that can be done to expand on this project.

1. Implement override function

   Currently the setup does not have an over ride function in case the user wants to shut down the system. Only a reset switch has been supplied. Furthermore, if a user wants to shut off lights or appliances despite being in the room, there currently is no way to do that.

2. Allow bi-directional communication

   Enhance the system to allow bi-directional communication between the sensor circuits and the control station. For example, if the MainMonitor detects that nobody is in the room, then a sleep feature for local sensors could be implemented. The sensors will wake up once someone enters the room.

3. Use antennas

   This project dealt with the simulation of wireless channels using the TIMS. However, true wireless could be performed by using antennas with the TIMS system. There are antenna modules from Emona that can be purchased to use with the system. A comparison between simulation and actual wireless transmission could be performed.

4. Use real wireless modules

   Using the TIMS as a guide, use wireless modules that use protocols such as Bluetooth or ZigBee to develop a wireless sensor network. This would increase flexibility and address the practical side of sensor networks more so than working solely with the TIMS.

5. Putting circuitry on printed circuit boards

   This would greatly improve manageability and aesthetics of the system product. The system would be more robust as well. Less loose wire is a good thing.

6. Multiple control stations or multi-room setup

   Implement the system with multiple control stations or in a multi-room configuration. Basically expand the system, its reach, and capabilities. Use something more powerful and flexible than the Altera Board.

7. Implement security features

   Detection of suspicious entry or activity. Proper reporting of such activity. Could interface with a computer to automatically file reports. Logs could also be kept.

7. Acknowledgements

I would like to thank Dr. Lynne A. Molter, my advisor, for her guidance and support throughout the course of this project. She was also very generous in allowing me to use a large section of her Optics and Photonics Laboratory as a workplace.

I would also like to thank Edmond Joudi for his advice, technical knowledge, and endless supply of electronic components and equipment.

Furthermore, I would like to take this opportunity to thank the Engineering Department Faculty and Staff for their guidance throughout my four years at Swarthmore College.

Finally, but not least, I would like to thank my family and friends for always being there for me.

8. References

Carlson, A. B., Crilly P. B., & Rutledge J. C. (2002). *Communication Systems: An Introduction to Signals and Noise in Electrical Communication*. New York, NY: Mc Graw Hill.

Hamblen, J. O., & Furman, M. D. (2001). *Rapid Prototyping of Digital Systems: A Tutorial Approach*. Norwell, MA: Kluwer Academic Publishers.

Tesfaye, A. & Whitehead, D. (2003). *Engineering Design Project Reports: Function-Specific Sensor Fusion*. Swarthmore, PA: Swarthmore College Engineering Department.

9. Appendix

VHDL Files

controlV5.vhd

```
-- Control Program for the Altera Board
-- 2 IR Sensors, 1 Pressure Sensors
-- Version 5
-- Code Written by Masabumi Chano
-- Last Updated: April 6, 2006

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE IEEE.std_logic_signed.ALL;

ENTITY ControlV5 IS
   PORT(  Reset  : IN std_logic;
      GClock : IN std_logic;
      InIRInner : IN std_logic;           -- Input from Inner IR Sensor
      InIROuter : IN std_logic;           -- Input from Outer IR Sensor
      InPressure1  : IN std_logic;        -- Input from Pressure Sensor
      ExtDisp1  : OUT std_logic_vector(6 DOWNTO 0);
      ExtDisp2  : OUT std_logic_vector(6 DOWNTO 0);
      OutMainLight : OUT std_logic;       -- Controls Main Light
      OutLocalLight1   : OUT std_logic);  -- Controls Local Light

END ControlV5;

-----------------------------------------------------------------

ARCHITECTURE Behavioral OF ControlV5 IS

-- Declare Components
   COMPONENT ClockDivider IS
      PORT(reset     : in  std_logic;
           clk       : in  std_logic;
           clockbits : out std_logic_vector(7 downto 0));
   END COMPONENT;
-- End Component Declaration

-- Declare Types and Signals
   TYPE MainStateType is (Snoop, EntryDetect, ExitDetect, EntryCount,
ExitCount);
   TYPE LocalStateType is (Idle, Active, SetOut, SetDelay);
   TYPE ControlType is (RelayOn, RelayOff, Countdown);
   TYPE TestType is (Idle, Active, Detect1, Detect2, Detect3, Detect4,
Test1);

   SIGNAL MainMonitor     : MainStateType;
   SIGNAL LocalMonitor    : LocalStateType;
   SIGNAL OutputMonitor   : LocalStateType;
   SIGNAL FlagP1          : ControlType;
   SIGNAL Test            : TestType;

   SIGNAL SlowClock       : std_logic;
```

```
    SIGNAL ResetButton     : std_logic;

    SIGNAL RelayML         : std_logic;
    SIGNAL RelayLL1        : std_logic;

    SIGNAL DisplaySequence : std_logic_vector(6 DOWNTO 0);
    SIGNAL ExtraDigit      : std_logic_vector(6 DOWNTO 0);

    SIGNAL Population      : std_logic_vector(2 DOWNTO 0); -- Counts pop.
    SIGNAL Timer1       : std_logic_vector(3 DOWNTO 0); -- Delay Timer
-- End Declarations

BEGIN
    PROCESS (ResetButton, GClock) BEGIN
        IF ResetButton = '1' THEN
            MainMonitor     <= Snoop;
            LocalMonitor    <= Idle;
            Population      <= "000";
            FlagP1          <= RelayOff;
            RelayML         <= '0';
            RelayLL1        <= '0';
            Timer1          <= "0000";
            DisplaySequence <= "1111110";
            ExtraDigit      <= "1111110";
            Test            <= Idle;

        ELSIF (GClock = '1' AND GClock'EVENT) THEN
            CASE MainMonitor IS
                WHEN Snoop =>
                    IF InIROuter = '1' THEN
                        MainMonitor  <= EntryDetect;
                    ELSIF  InIRInner = '1' THEN
                        MainMonitor  <= ExitDetect;
                    END IF;
                WHEN EntryDetect =>
                    IF InIRInner = '1' THEN
                        MainMonitor  <= EntryCount;
                    END IF;
                WHEN ExitDetect =>
                    IF InIROuter = '1' THEN
                        MainMonitor  <= ExitCount;
                    END IF;
                WHEN EntryCount =>
                    IF InIRInner = '0' THEN
                        Population   <= Population + 1;
                        MainMonitor  <= Snoop;
                    END IF;
                WHEN ExitCount =>
                    IF InIROuter = '0' THEN
                        Population   <= Population - 1;
                        MainMonitor  <= Snoop;
                    END IF;
            END CASE;

            CASE LocalMonitor IS
                WHEN Idle =>
                    CASE Population IS
                        WHEN "000" =>
                            DisplaySequence  <= "0000001";    -- No Lights
                            LocalMonitor     <= Idle;         -- Nobody in rooom
                        WHEN "001" =>
                            DisplaySequence  <= "1001111";
                            LocalMonitor     <= Active;
```

```
                  WHEN "010" =>
                     DisplaySequence  <= "0010010";
                     LocalMonitor     <= Active;
                  WHEN "011" =>
                     DisplaySequence  <= "0000110";
                     LocalMonitor     <= Active;
                  WHEN "100" =>
                     DisplaySequence  <= "1001100";
                     LocalMonitor     <= Active;
                  WHEN "101" =>
                     DisplaySequence  <= "0100100";
                     LocalMonitor     <= Active;
                  WHEN "110" =>
                     DisplaySequence  <= "0100000";
                     LocalMonitor     <= Active;
                  WHEN OTHERS =>
                     DisplaySequence  <= "0001111";
                     LocalMonitor     <= Active;
               END CASE;
            WHEN Active =>
               IF  InPressure1 = '1' THEN
                  LocalMonitor        <= SetOut;
               ELSIF InPressure1 = '0' THEN
                  CASE FlagP1 IS
                     WHEN RelayOn =>
                        LocalMonitor  <= SetDelay;
                     WHEN Countdown =>
               --        Timer1       <= Timer1 + 1;
                        LocalMonitor  <= Idle;
                     WHEN OTHERS =>
                        LocalMonitor  <= Idle;
--                      LocalMonitor  <= SetTimer;
                  END CASE;
               END IF;
            WHEN SetOut =>
               FlagP1                 <= RelayOn;
               LocalMonitor           <= Active;
            WHEN SetDelay =>
               FlagP1                 <= Countdown;
               LocalMonitor           <= Idle;
         END CASE;

         --------------------------------
         -- OUTPUTS AND TIMER CONTROLS --
         --------------------------------
         ExtraDigit <= "0000001";
         CASE OutputMonitor IS
            WHEN Idle =>
               CASE Population IS
                  WHEN "000" =>
                     OutputMonitor<= Idle;     -- Nobody is in the rooom
                     RelayML      <= '0';      -- Relays are OFF
                     RelayLL1     <= '0';
                     Timer1       <= "0000";
                     FlagP1       <= RelayOff;
                  WHEN OTHERS =>
                     OutputMonitor<= Active;
               END CASE;
            WHEN Active =>
               RelayML      <= '1';
               CASE FlagP1 IS
                  WHEN RelayOn =>
                     Timer1    <= "0000";
```

```
                        RelayLL1  <= '1';          -- RelayLL1 IS ON
                     WHEN Countdown =>
                        CASE Timer1 IS
                           WHEN "1111" =>
                              FlagP1    <= RelayOff;
                           WHEN OTHERS =>
                              Timer1    <= Timer1 + 1;
                        END CASE;
                     WHEN RelayOff =>
                        RelayLL1  <= '0';          -- RelayLL1 IS OFF
                        Timer1       <= "0000";
                  END CASE;
                  OutputMonitor<= Idle;
               WHEN OTHERS =>
                  OutputMonitor<= Idle;
            END CASE;
         END IF;
      END PROCESS;

      ClockIt  : ClockDivider PORT MAP (ResetButton, GClock, SlowClockBits);--
      ResetButton  <= NOT Reset;
      MainClock <= SlowClockBits(7);

      ExtDisp1  <= ExtraDigit;
      ExtDisp2  <= DisplaySequence;

      OutMainLight <= RelayML;
      OutLocalLight1   <= RelayLL1;

END Behavioral;
```

clockdivider.vhd

```
-- Clock Divider
-- Adapted from Prof. Maxwell's clockdivider.vhd

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
USE ieee.std_logic_unsigned.ALL;

ENTITY ClockDivider IS
   PORT(Reset       : IN std_logic;
        CLK         : IN std_logic;
         ClockBits  : OUT std_logic_vector(7 DOWNTO 0));
END ClockDivider;

ARCHITECTURE Behavioral OF ClockDivider IS
   SIGNAL Counter   : unsigned(22 DOWNTO 0);      -- big counter
BEGIN
   PROCESS (CLK) BEGIN
      IF Reset = '1' THEN
         Counter <= "00000000000000000000000";   -- reset the clock
      ELSIF (CLK = '1' AND CLK'EVENT) THEN
         Counter <= Counter + 1;
      END IF;
   END PROCESS;
   ClockBits <= std_logic_vector(Counter(22 DOWNTO 15));
END Behavioral;
```

testLight.vhd

```
-- On/Off Program
-- Testing the IR Sensors
-- Code Written by Masabumi Chano
-- Last Updated: Feb 9, 2006

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE IEEE.std_logic_signed.ALL;

ENTITY testLight IS
   PORT(  CLK          : IN std_logic;
          Reset        : IN std_logic;
          GClock       : IN std_logic;
          SensorIn     : IN std_logic;
          ToLight      : OUT std_logic;
          ShowReset    : OUT std_logic;
          ToRelay      : OUT std_logic);
END testLight;

---------------------------------------------------------------------

ARCHITECTURE Behavioral OF testLight IS

-- Declare Components
   COMPONENT ClockDivider IS
      PORT(reset     : in  std_logic;
```

```
            clk       : in  std_logic;
            clockbits : out std_logic_vector(7 downto 0));
   END COMPONENT;
-- End Component Declaration

-- Declare Types
   TYPE stateType is (ReadIn, SetOut);
   TYPE relayType is (RelayOn, RelayOff);
   SIGNAL State        : stateType;
   SIGNAL RelayState   : relayType;
-- End Type Declaration

-- Declare Signals
   SIGNAL SensorState     : std_logic;
   SIGNAL SlowClock       : std_logic;
   SIGNAL SlowClockBits   : std_logic_vector(7 DOWNTO 0);
   SIGNAL ResetButton     : std_logic;
   SIGNAL MainClock       : std_logic;
   SIGNAL Flag            : std_logic;
-- End Signal Declaration

BEGIN
   PROCESS (ResetButton, MainClock) BEGIN
      IF ResetButton = '1' THEN
         ShowReset    <= '1';
         State        <= ReadIn;
         Flag         <= '0';
         RelayState   <= RelayOff;
         ToLight      <= '0';
         ToRelay      <= '1';
      ELSIF (MainClock = '1' AND MainClock'EVENT) THEN
         CASE State IS
            WHEN ReadIn =>
               ShowReset       <= '0';
               IF SensorIn = '1' AND Flag = '0' THEN
                  State        <= SetOut;
                  Flag         <= '1';
               ELSIF SensorIn = '0' THEN
                  State        <= ReadIn;
                  Flag         <= '0';
               END IF;
            WHEN SetOut =>
               State           <=ReadIn;
               IF RelayState = RelayOn THEN
                  ToRelay      <= '1';
                  ToLight      <= '1';
                  RelayState   <= RelayOff;
               ELSIF RelayState = RelayOff THEN
                  RelayState   <= RelayOn;
                  ToLight      <= '0';
                  ToRelay      <= '0';
               END IF;
         END CASE;
      END IF;
   END PROCESS;

   ClockIt  : ClockDivider PORT MAP (ResetButton, GClock, SlowClockBits);
   ResetButton <= NOT Reset;
   MainClock   <= SlowClockBits(7);

END Behavioral;
```