DESIGN AND IMPLEMENTATION OF BUILDING AUTOMATION SYSTEM OVER POWER LINE NETWORK

By

MUAAZ ABBAS HUSAIN A.ALLAH

INDEX NO. 044069

Supervisors

Ust. Loui Faroug
Ust. Samah Hashim

A report submitted in partial fulfillment of the requirements for the degree of
B.Sc. (HONS)
In
Electrical and Electronic Engineering
To
Department of Electrical and Electronic Engineering
Faculty of Engineering and Architecture
University of Khartoum

July 2009
إهداء

إلى كل العاملين في شتى بقاع الأرض ....

بصدق واخلاص ....

لأجل رفعة الأمة ...

بالرغم من طول الطريق ...

إليهم أهدي جهدي المتواضع ...

عسى أن يكون خطوةً على ذات الطريق ...

معاذ عباس حسين

يوليو 2009
ACKNOWLEDGEMENT

All praise and thanks is due to Almighty God, the creator and sustainer of the worlds. I wish to thank Him for all that He has gifted us with, although, He can never be praised or thanked enough.

I would like to express my deepest sense of gratitude to my supervisors, Ust. Loui Faroug & Ust. Samah Hashim for their continuous advice, systematic guidance, encouragement and great efforts through the course of this project.

I am deeply grateful to my project partner, Ghanim RidaEldein for his hard work, real team spirit, continuous support, the unforgettable times we have spent and all good things I have learned from him.

I have greatly profited from hints, discussions and experiences generously lavished during the period of my university study from my friends and classmates. I would like to specially thank Hatim Almoala, Osman Abdallah and Abdualteef Hassan for the vivid knowledge-sharing environment I have experienced with them.

I also would like to give my gratitude to my lovely residence mates in Abodijana for the five years period with laughter, dreams, mutual encouragement and love we had.

On a more personal note I would like to express my love and gratitude from my deep heart to my parents Abbas Husain and Mahasin Husain and my brothers for teaching me values that are precious, irrespective of time and place and for their endless love, support and constant encouragement in all my endeavors.

I am deeply grateful for all people who encouraged and supported me during my university study.
ABSTRACT

Building automation is now very widespread in all the developed countries to ease the management of buildings like schools, hospitals, public edifices, private houses and so on.

In this project, a feasible Building Automation System based on very cheap distributed microcontroller architecture was designed. A power line communication network was established to link those microcontrollers; so there is no need to install proprietary cables or a wireless network.

The Power Line Communication (PLC) technology utilizes the power cables as a communication channel. A radio frequency signal of a few hundreds of Hz up to a few tens of MHz is transmitted through the power lines. The power line communication has distance and frequency limitations which depend on the type of power line used. The device performs the operation of data transfer through the power line and connects controllers to the power lines; is called the Power Line Modem (PLM).

An illustrative model was designed and implemented to clarify the idea of Building Automation Over Power Line, this model contains a user interface, air conditioning system and lighting system; those systems are connected through the power line network.

المستخلص:

عملية اتّخاذ المباني واسعة الانتشار في كثير من البلدان المتطورة بغرض تيسير ادارة المباني مثل المدارس، المستشفيات، المراقب العامة والمنازل. في هذا المشروع تم تصميم نظام تحكم في المبنى مستندًا للمتحكمات الدقيقة الموزعة ذات الأسعار الزهيدة، كما تم تأسيس شبكة تحكم عبر خطوط الكهرباء لربط تلك التحكمات الشيء الذي يغني عن تثبيت كابل خاصة أو شبكة تحكم لاسلكية.

إن تكنولوجيا الاتصال عبر خطوط الكهرباء تستخدم كوابلك الكهرباء كقناة اتصال؛ فإشارة راديو بتردد يساوي مئات الهيجلز حتى عشرات الميغاهرتزات يتم نقلها من خلال خط الكهرباء. إن تقنية الاتصال عبر خطوط الكهرباء لها تقنيات للمسافة والأمران والتي تعتمد على نوع خط الكهرباء المستخدم. فالجهاز الذي يقوم بعملية نقل البيانات عبر خطوط الكهرباء ويربط التحكمات بخط الكهرباء يسمى بمودم الاتصال عبر خطوط الكهرباء.

تم تصميم وتنفيذ نموذج توضيحي لتوضيح فكرة التحكم في المباني من خلال خطوط الكهرباء، هذا النموذج يحتوي على: واجهة المستخدم، نظام تكييف ونظام الإضاءة، كل هذه الأنظمة موصولة عبر خط الكهرباء.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>TITLE</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>إهداء</td>
<td>II</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td>III</td>
</tr>
<tr>
<td>Abstract</td>
<td>IV</td>
</tr>
<tr>
<td>المستخلص</td>
<td>IV</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>V</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>X</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>XIII</td>
</tr>
</tbody>
</table>

**Chapter 1: INTRODUCTION**

1. Overview ................................................................. 1
2. Statement of the problem ............................................ 1
3. Project motivations ................................................. 2
4. Project Objective .................................................... 2
5. Thesis Layout ......................................................... 2

**Chapter 2: LITERATURE REVIEW**

1. Building Automation Systems (BAS) .............................. 3
   1.1 Definition .............................................................. 3
   1.2 Benefits .............................................................. 3
   1.3 Typical BAS architecture ......................................... 3
   1.4 Main concepts of BAS ............................................. 4
2. Power Line Communication (PLC) .................................. 6
   2.1 Definition ............................................................. 6
   2.2 Concept ............................................................... 6
   2.3 Types of PLC frequency bands .................................. 7
Chapter 3: METHODOLOGY AND DESIGN

3.1 Design Goals

3.1.1 User convenience and system portability

3.1.2 Hardware independence

3.1.3 System integration

3.2 System requirements

3.2.1 Establishment of a Power line communication network

3.2.2 Design of simple illustrative modules

3.2.3 Monitor the BAOPL system on PC

3.3 System Layout

3.4 System scenario

3.4.1 Air conditioning system

3.4.2 Lighting control system

3.4.3 User Interface module

3.5 Development Tools

3.5.1 Hardware tools

3.5.2 Software Tools

3.5.2.1 MicroSoft Visual Basic 6 (MS VB6)

3.5.2.2 CCS compiler software

3.5.2.3 MicroPro-USB PIC programmer software

3.5.2.4 Proteus7.1

3.6 Design of System components

3.6.1 Power Line Modem (PLM)

3.6.2 PLC-UART/RS232 Power line modem

3.6.3 RS-232 Communication protocol

3.6.4 USART Module
4.6.1 MCU programming development.................................................. 61
4.6.2 Hardware development.............................................................. 61
  4.6.2.1 Brief description of required hardware components................. 61
  4.6.2.2 Hardware connection......................................................... 62
  4.6.2.3 System packaging......................................................... 62
4.7 The results..................................................................................... 64

Chapter 5: CONCLUSION AND FUTURE DEVELOPMENTS 66

  5.1 Conclusion................................................................................. 66
  5.2 Problems and solutions.............................................................. 67
    5.2.1 Broadcasting communication.............................................. 67
    5.2.2 Single phase restriction...................................................... 67
  5.3 Future developments.................................................................. 68
    5.3.1 Outdoor accessibility............................................................ 68
    5.3.2 Building Management Over Power Line.............................. 68
    5.3.3 Other applications of power line communication.................. 68

References.......................................................................................... 69

Appendix A: CIRCUIT DIAGRAMS...................................................... A-1
Appendix B: PROJECT CODES.............................................................. B-1
LIST OF FIGURES

Figure2.1 Typical BAS architecture................................................................. 4
Figure2.2 Example of refrigerator controller.................................................... 5
Figure2.3 Power line channel spectrum ........................................................... 7
Figure2.4 Home networking over power line................................................... 9
Figure2.5: Layout of Building Automation Over Power lines......................... 9
Figure2.6 An illustrative diagram for some PLC applications........................ 10
Figure2.7 Noise in PLC environment.............................................................. 11
Figure2.8 Spectrum diagram of modulated signal over PLC channel.............. 12
Figure2.9 Binary ASK diagram........................................................................ 13
Figure2.10 Binary FSK/PSK modulation process.............................................. 14
Figure2.11 Coherent and Non-Coherent Digital modulators............................ 15
Figure2.12 Layout of PLM connection............................................................... 16
Figure2.13 Typical architecture of the PLM system.......................................... 17
Figure2.14 Power line coupler ....................................................................... 18
Figure2.15 Typical diagram of band pass filter................................................. 18
Figure2.16 A generic view of a microcontroller............................................... 20
Figure2.17 Microcontroller Architecture......................................................... 21
Figure2.18 Applications of Microcontrollers................................................... 22
Figure 3.1 BAOPL system block diagram

Figure 3.2 Event handling in Visual Basic

Figure 3.3 Functional Diagram of PLC-UART/RS232

Figure 3.4 RS-232 frame format

Figure 3.5 Description of BAOPL protocol

Figure 3.6 Layout of Air conditioner system

Figure 3.7 Firmware of Air Conditioning system: Flow chart of MCU

Figure 3.8 Firmware of Air Conditioning system: Flow chart of USART ISR

Figure 3.9 Firmware of Air Conditioning system: Flow chart of Timer0 ISR

Figure 3.7 Layout of Lighting control system

Figure 3.8 Firmware of Lighting control system: Flow chart of MCU

Figure 3.9 Firmware of Lighting control system: Flow chart of USART ISR

Figure 3.10 HMI module: Firmware flow chart

Figure 3.11 HMI module: Flow chart of COM1 Receive ISR

Figure 3.12 HMI module: Flow chart of COM1 Receive ISR

Figure 3.13 HMI module: Flow chart of user ISR

Figure 4.1 Snapshot of HMI interface

Figure 4.2 PIC16F877 pinout

Figure 4.3 Snapshots of the PLC-UART/RS232/RS485 Transceiver Module

Figure 4.4 Photo sensor (LDR sensor) appearance

Figure 4.5: IR Receiver pinout
Figure 4.6 LM555 timer pinout and connection…………………………………… 58
Figure 4.7 ULN2003 pinout……………………………………………………………… 58
Figure 4.8 Lighting control system after implementation………………………… 60
Figure 4.9 The packaged model of lighting control system…………………… 60
Figure 4.10 LM35 pinout……………………………………………………………… 61
Figure 4.11 The Air conditioner control module after implementation …….. 63
Figure 4.12 The packaged model of Air conditioning system………………….. 63
Figure A.1 simulation circuit for the lighting control system………………….. A-1
Figure A.2 Simulation circuit for air conditioner subsystem………………….. A-2
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAS</td>
<td>Building Automation System.</td>
</tr>
<tr>
<td>PLC</td>
<td>Power Line Communication.</td>
</tr>
<tr>
<td>bps</td>
<td>bit per second (data rate)</td>
</tr>
<tr>
<td>PLM</td>
<td>Power Line Modem</td>
</tr>
<tr>
<td>USART</td>
<td>Universal Synchronous Asynchronous Receiver Transmitter</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to Digital Converter</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to Analog Converter</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>HMI</td>
<td>Human Machine Interface</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro-Controller Unit</td>
</tr>
<tr>
<td>FSK</td>
<td>Frequency Shift Keying</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1 Overview

Building Automation System is a network of controllers that controls the climate in the building and supervises the building facilities heating by integration of building systems such as security, fire alarm, lighting control, etc. The greatest benefit behind using Building Automation Systems is the allowance of preventive measures to be taken before equipment failure. Building automation is now very widespread in all the developed countries to ease the management of buildings like schools, hospitals, public edifices, private houses and so on.

In this project, a feasible Building Automation System based on very cheap distributed microcontroller architecture was designed. A power line communication network was established to link those microcontrollers; so there is no need to install proprietary cables or a wireless network.

The Power Line Communication (PLC) technology utilizes the power cables as a communication channel. A radio frequency signal of a few hundreds of Hz up to a few tens of MHz is transmitted through the power lines. The power line communication has distance and frequency limitations which depend on the type of power line used. The device performs the operation of data transfer through the power line and connects controllers to the power lines; is called the Power Line Modem (PLM).

An illustrative model was designed and implemented to clarify the idea of Building Automation Over Power Line, this model contains a user interface, air conditioning system and lighting system; those systems are connected through the power line network.

1.2 Statement of the problem

The traditional Building Automation Systems use proprietary cables to connect between the system terminals, usually the installation of these cables yields a high cost and nonflexible network, also repeated modifications might be made for system expansions. Major of buildings have a ubiquitous power line that could be accessed any where inside
the building, therefore, a feasible solution is introduced in this project, this solution is a Building Automation Over Power Line System.

1.3 Project Motivations

- Convenience of installation and modification of the building automation systems, especially for old and not pre-automated buildings.
- Accessibility of automation system any where inside a building.
- Eliminate the installation costs of buildings networking.
- Convenient user accessibility to the building automation system; just has to plug and play.

1.4 Project Objectives

- Study of the possibility and feasibility of power line communication as a solution for building networking.
- Designing and establishing a control network base on power line communication technology.
- Design and implementation of a feasible building automation system based on distributed microcontroller architecture.

1.5 Thesis Layout

This thesis is organized as follows:

Chapter 2: Introduces the concepts of the building automation systems and power line communication including their requirements.

Chapter 3: The methodology and design, architecture and implementation of the various parts of our system are comprehensively explained.

Chapter 4: Describes the procedures and experiments carried out to evaluate system’s performance then presents the experimental results deduced from these experiments.

Chapter 5: Presents conclusions and recommended future work.

Appendix A: presents the hardware circuit diagrams.

Appendix B: Gives system programming codes.
2. LITERATURE REVIEW

This chapter reviews the current literature and related topics of Building Automation Systems, Power line communication principles and design, and finally the Microcontrollers. Through examination of various online resources, journals, and texts, the necessary background information for the rest of this project has been attained.

2.1 Building Automation Systems (BAS)

2.1.1 Definition

BAS is a network of controllers that manages the building facilities instead of human being, such as ventilation, lighting, security, supervision, etc. [1]. Therefore BAS is designed to perform various tasks, such as:

- Building climate Control.
- Supervision and control of heating, ventilation, and air conditioning equipment.
- Facility management (generates alarms and reports when there is a problem).
- Energy management strategies to reduce operating and energy costs.
- Integrate building systems such as security, fire alarm, lighting control, an so on.

2.1.2 Benefits

Benefits are concluded in the following points:

- Allows building managers to monitor and control one or many facilities either on-site or remotely saving time and money.
- Improves service diagnostics and allows preventive measures to be taken before equipment failure.
- Reduces comfort complaints from tenants/employees.
- Improves staff minimization by equipments automation.
- Integrates the key facility systems and control devices.
- The electrical energy can be managed, so it reduces energy use.
- Reduces the operating time of equipment as less as possible, so it increases equipment efficiency and life time.

2.1.3 Typical BAS architecture

The typical layout of BAS is represented below:
2.1.4 Main components of BAS

1. The user interface:

   The purpose of user interface is the interfacing between the human and the hardware of the system, so it exposes the system parameters to the user in a suitable format graphically (graphical interface) or in a form of text (text display). Sometimes user interface is called Human Machine Interface (HMI).

   The main tasks of user interface are monitoring the system parameters/variables, issuing the user format and providing away to access the settings of embedded systems.

   Usually, the user interface is the master controller in the system and general purpose computer is most widely used type of graphical interfaces due to its suitability to domestic usage, upgradability, flexibility and portability. Sometimes embedded LCD screen is used instead of computer.[1].

2. The terminal controllers:

   The terminology of “terminal controllers” means those controllers used to help the master controller to control the entire system, usually they work as slave controllers listen to requests of the master controller, accomplish his wants and reply his questions. Another important task of slave controllers is to hold the burden of control processes instead of the master controller, therefore majority of BASs work in manner of distributed control system.

![Figure2.1 Typical BAS architecture][3]
Microcontrollers and Programmable Logic Controllers are used as slave controllers, because they are dedicated for this purpose in-addition to their ability to communicate with other digital systems such as computers.

3. Control equipments:

   They are equipments used by the slave controller “to see and change the world”, these equipments include:
   - Detection devices which are used to detect either presence or absence of a physical phenomenon, such as light detectors, smoke detectors, etc.
   - Measuring devices which are used to convert a physical phenomenon to a scalable electrical signal, such as temperature sensor, humidity sensor, wind speed sensor, and so on.
   - Indication devices which are used to indicate status of controlled devices, such as indication lamps, displays, etc.
   - Switching devices which are used in switching processes, such as pushbuttons, selective switches, etc.
   - Coupling devices which are used in coupling among different level electrical circuits, such as relays, contactors, etc.
   - The controlled device which is the device to be controlled, such as lamps for lighting control system, fans for ventilation control system, and so on.
4. The control network:

   The control network of BAS is a network that links the master controller (User Interface) with the slave controllers; therefore it provides data means to exchange the control data (commands, error messages, requests, etc.) between the nodes.

   Up to now days, all of BAS’s networks are done using proprietary techniques which require special wiring (such as twisted pairs, coaxial and fiber optic) or special wireless networking (eg. Bluetooth or Home RF). [2].

   An economical and feasible solution is to use the pre-installed household power line instead of those proprietary solutions. According to the Power Line Communication technology; the household power line could be used as a BAS network. The Power Line Communication technology is discussed in the following section.

2.2 Power Line Communication (PLC)

2.2.1 Definition

   Power line communication is “the technology which provides a way to send and receive voice and data signals through the existing power line network”. [4]. Therefore it is a system for carrying data on a conductor that is also used for electric power transmission.

   Power Line Communication (PLC), also is known as Power-line Digital Subscriber Line (PDSL), Power Line Telecom (PLT), or Power Line Networking (PLN). [3]

2.2.2 Concept

   Power line communication system operates by impressing a modulated carrier signal into the power line wiring and extracting such a signal from the same power line. Different types of power line communication use different frequency bands, depending on the signal transmission characteristics of the power wiring used. Since the power wiring system was originally intended for transmission of AC power, the power line wire has limitations for carrying high frequencies. The propagation problem is a limiting factor for each type of power line communications.

   Most of world power systems use 50Hz or 60Hz AC power (50Hz in Sudan). The power line channel spectrum is shown below.
AC power lines are used as a medium to send and receive discrete frequency-based control, monitoring and communication messages to run building devices, exchange data and share information among multiple PCs and other devices. Basically a radio signal is modulated with the data that has to be sent. This radio signal is then sent down the copper medium (selected power lines) in a band of frequencies not used for the purpose of supplying electricity. The frequencies and encoding schemes used greatly influence both the efficiency and the speed of the PLC service.

Data rates over a power line communication system vary widely. Low-frequency (about 100-200 kHz) carriers impressed on high-voltage transmission lines with an equivalent data rate of a few hundreds bps. Higher data rates generally imply shorter ranges; a local area network operating at millions of bps may only cover one floor of an office building, but eliminates installation of dedicated network cabling.

2.2.3 Types of PLC frequency bands

- High frequency band PLC ($f \geq 1\text{MHz}$):

  High frequency communication may use large portions of the radio spectrum for communication, depending on the technology. This type is applied for low-voltage power lines, so it is dedicated to household usage.

  High frequency PLC can be used to interconnect home computers, peripherals or other networked consumer peripherals, and internet access broadband over power lines. [5].

- Medium frequency band PLC ($\text{kHz}<f<1\text{MHz}$):
This is the most famous application which uses the electrical power wiring as a transmission medium. It suites the low/medium voltage power line, so it is a suitable technique for building automation that provides remote control of lighting and appliances without installation of additional control wiring.[5]

- Low frequency band PLC (f<1KHz):
  This type suites the high voltage transmission lines, so it can be used in power metering and transmission of supervisory data in low data rate (up to hundreds of bps).[5]

2.2.4 Advantages of PLC

From the pervious section, obviously, it is possible to use the existing household power line instead of proprietary wiring techniques. In this section some of PLC advantages are discussed to show the advantage of PLC over other techniques used in BASs:

- Usage of existing power line wiring leads to save money and efforts of a new communication infrastructure installation.
- High flexibility; any power line socket/node can be used to plug and play.
- Easy to install and modify; doesn’t need high experience.
- Economically feasible; it saves the additional costs of network installation.
- Has acceptable range, up to 800m in some types and can be extended to be infinite by using repeaters.
- Fast data transmission rate up to 10Mbps.

2.2.5 General Applications of PLC

The most applications which utilize PLC are summarized in the following points:

- PLC is used to provide some domestic services such as home networking, consumer peripherals interconnection, and broadband Internet access.
Building automation and management:

Typically home-control power line communication devices operate by modulating in a carrier wave of between 20 and 200 kHz into the household wiring at the transmitter. The carrier is modulated by digital signals. Each receiver in the system has an address and can be individually commanded by the signals transmitted over the household wiring and decoded at the receiver. These devices may be either plugged into regular power outlets, or permanently wired in place. Since the carrier signal may propagate to nearby homes (or apartments) on the same distribution system, these control schemes have a "house address" that designates the owner.[5]
• Automation of vehicles using its existing DC power network.
• Automated Meter Reading (AMR) devices.
• Remote observation of substations using the existing power’s transmission lines.
• Telecommunications in rural areas which are connected to the general electricity network.
• Outdoor applications like control of public streets lamps.
• Industrial applications, like the control of machines through the electric feeding wires.
• Transmission of information of CCTV cameras used in security systems.

![An illustrative diagram for some PLC applications](image)

2.2.6 Drawbacks of PLC and their solutions

• Noisy environment:

  The electrical wires are not originally dedicated for communication, so it is a noisy medium. In addition of noisy medium, some of building appliances generate noise signals, such as: Brush motors, bulbs and lamps, switching power supplies, power tools etc.

  Therefore it is very important to take in mind this drawback and attempt to reduce the effect of noise as less as possible.
- **Impossibility of direct connection**

  It is impossible to exchange messages directly through the power line medium, because of existence of the live power (e.g., 220V/50Hz) which is impossible to be interfaced to the digital/DC systems; this problem has been solved by development of such device which is called “Power Line Modem (PLM)”, PLM is a digital transceiver which enables transmission and receiving of digital data through power line.[3]

- **Range limitations**

  The PLC data rate depends mainly on the range to be used for, so the data rate could be higher in small ranges rather than large ones. But improved encoding techniques can help to rectify it.

- **Sharing of bandwidth.**

  Each terminal of BAS network requires significant data rate. However the amount of bandwidth available through the network must be shared fairly and consistently. This problem can be rectified using suitable high performance modulation techniques like: FSK, PSK and OFDM.

- **Security of data**

  PLC network, as any other network, needs to be secured and protected from unauthorized access; therefore encryption must be used to prevent the interception of sensitive data by unauthorized personnel.

- **Interference**

  Electrical wires can pickup the RF signals, thus other types of RF signals may interfere with PLC signal of the same frequency. So it is necessary to use unused bandwidths.
2.2.7 Stages of PLC process

2.2.7.1 Interfacing to the power lines.

Due to incompatibility between the AC power (220V/50Hz or more) and the digital components of controllers; the interfacing process is a critical issue.

The power line coupler helps to connect the low voltage digital devices to the power line without damaging them. It also acts as a high-pass filter to stop power signal. It allows injection and acquisition of communication signals from and to the Power line [8]. Therefore the converted signal could be manipulated using Digital Signal Processing techniques.

2.2.7.2 Filtering

The control signal and power are transmitted together through the same medium, but they are different in shape, bandwidth and frequency. Obviously it is not true to manipulate the mixed signals, so an important stage of signals separation must be done. Band Pass Filter is the suitable choice to do this function [7]. The acquired signal is filtered to remove any high frequency noises because it can pass only the bandwidth of control signal and eliminate power signal.

2.2.7.3 Modulation

Modulation is needed to use the uncovered bandwidths of power line channel spectrum as shown in figure 2.9 below:

![Figure 2.8 Spectrum diagram of modulated signal over PLC channel](image)

Generally, there are many techniques for modulation. All practical modulation techniques use the sinusoidal signal as a carrier:

\[ \text{carrier} = A \sin(2\pi f + \varphi) \]
The sinusoidal signal has three attributes that could be changed to represent the message:

- Amplitude (amplitude modulation).
- Frequency (frequency modulation).
- Phase (phase modulation).

Since all of modern BASs are based on digital systems, so it is enough only to study the digital modulation techniques. Digital modulation maps all data signal variations (0, 1) into one of the carrier attributes.

The major techniques of digital modulation are:

1. Binary Amplitude shift keying (ASK)

![Figure 2.9 Binary ASK diagram](image)

Obviously, the ASK technique is not immune to channel noise while the power line network is a higher noisy environment. So the ASK technique is not a proper modulation choice.

2. Binary Frequency shift keying and Phase Shift Keying (FSK and PSK)

Here the data variations are mapped into the frequency (or phase angle) of the signal. The frequency (or phase angle) has two distinct values that represent (0,1)
say \((f_1/\Phi_1\) and \(f_2/\Phi_2\)). The following equation and figure illustrate this technique.

\[
FSK(t) = \begin{cases} 
\sin(2\pi f_1 t) \text{ for bit 1} \\
\sin(2\pi f_2 t) \text{ for bit 0}
\end{cases}
\]

Figure 2.10 Binary FSK/PSK modulation process. [2]

a- Typical FSK/PSK modulation  
b- FSK signal representation  
c- PSK signal representation
As a conclusion, both of FSK and PSK are immune to channel noise, and have high performance; therefore they are used in PLC.

2.2.7.4 Demodulation

Demodulation is the opposite process of modulation, while the digital modulation is used in PLC, therefore the corresponding digital demodulation techniques are:

- Coherent demodulation: a technique needs phase information.
- Non coherent demodulation: doesn’t need phase information.

The following block diagrams represent the process for a coherent and non coherent demodulation.

![Block Diagrams](image)

Figure2.11 Coherent and Non-Coherent Digital modulators

Generally, there is no difference between performance of coherent and non-coherent demodulators and both of them are used in the PLC.

2.3 Power Line Modem (PLM) Design

2.3.1 Overview

PLM is a device that accomplishes the data transfer through the power line, impresses transmitted signal into power line and extracts received signal from it. The main function of Power line modem is to interface digital devices (computers, Microcontrollers, etc.) to the power line. [8]
2.3.2 Design guidelines of simple and low data rate PLM

2.3.2.1 General PLM architecture:

Figure 2.14 illustrates the architecture of the PLM used to achieve low rate digital data transmission and reception, displays the functional hierarchy of different modules. Broadly, the modules can be categorized into the followings:

- Power Line Coupler.
- Analog and Digital Signal Processor.
- Modulator and Demodulator.
- Line Driver.
- Interface.

![Diagram of PLM connection](image)
The coupler allows injection and acquisition of communication signal from and to the power line. The acquired signal is filtered to remove any high frequency noises and is amplified; the operation is performed by the analog signal processing unit. After the analog signal processor, the signal is passed through demodulator to extract the information. The output of the demodulator is passed through digital signal processing unit module for the reconstruction of the information signal. Similarly, at the transmitting part, the information is modulated, passed through the line driver and then injected to the power line.

2.3.2.2 Power line Coupler:

To be able to design a complete PLC system, a proper interface unit between relatively high voltage power line voltage (HV) and low voltage (LV) communication system, isolation of LV communication circuit from HV power line with appropriate frequency response to allow message signal injection and acquisition are needed to be
considered. High voltage rating coupling capacitors and isolation transformer were used for constructing the Power Line Coupler, which is shown in Fig. 2.14 below.

![Figure 2.14 Power line coupler](image)

**2.3.2.3 Analog Signal Processing Unit:**

The received signal from coupler was first sent through active band pass filter. Stage of amplification, bi-quad filter was used to provide adequate attenuation of power signal and other high frequency noises.

The band-pass filters were designed to have center frequencies of carrier signals.

![Figure 2.15 Typical diagram of band pass filter](image)

**2.3.2.4 Digital Signal Processing Unit**

The main purpose of the digital signal processing is ‘Signal Reconstruction’ using special DSP chips to extract the acquisitioned data and reconstruct the original waveform.

**2.3.2.5 Modulation and demodulation:**

As mentioned in section 2.2.7.3 and 2.2.7.4, the suitable digital modulation/demodulation techniques are used in PLC. The signal is passed through demodulator to extract the raw data to be used by the DSP unit. On the other hand the
information signals are modulated to match the noisy electric grid environment, passed through the line driver and to then it is injected to the power line.

2.4 Microcontrollers

2.4.1 Introduction:

Microcontroller is a computer on a chip that is programmed to perform any control, sequencing, monitoring and display function. Because of its relatively low cost and programming simplicity, it is a good choice for digital control system design.[9]

Here the need was not necessarily for high computational power, or huge quantities of memory, or very high speed. Primarily, it must have excellent input/output capability, and be able to operate with the harsh conditions of the industrial and extremes of temperature.

2.4.2 General features of microcontrollers:

In addition to the CPU, RAM and ROM, a single chip microcontroller includes additional built-in hardware that makes it easier to implement a variety of embedded systems requirements. Like any electronic circuit the microcontroller needs to be powered, and needs a clock signal to drive the internal logic circuits. A typical microcontroller may include the following features:

- Timers and counters.
- Universal synchronous Asynchronous Receiver Transmitter (USART).
- Analog to Digital Converters (ADC).
- I/O ports.
2.4.3 Microcontrollers Architecture:

1. von-Neumann architecture:

The von Neumann architecture is a computer design model that uses a processing unit and a single separate storage structure to hold both instructions and data. It is named after mathematician and early computer scientist John von Neumann. Such a computer implements a universal Turing machine, and the common "referential model" of specifying sequential architectures, in contrast with parallel architectures.

2. Harvard architecture:

Harvard architecture is a newer concept than von-Neumann. It is a computer with physically separate storage and signal pathways for instructions and data. The term originated from the Harvard Mark I relay-based computer, which stored instructions on punched tape (24 bits wide) and data in electro-mechanical counters (23 digits wide). These early machines had limited data storage, entirely contained within the data processing unit, and provided no access to the instruction storage as data, making loading and modifying programs an entirely offline process.
2.4.4 Programming of Microcontrollers:

A microcontroller continuously executes the program stored in its memory. Assembly language is usually used to wire microcontrollers codes. For more complicated programs, C compilers are available to program, a variety of microcontrollers language offers more option to the programmers since it contains many functions and more complex data types. C codes also provide header files for many chips and libraries for common hardware used; this hides much of the hardware complexity from the programmer. Assemblers are used to convert the C code into a HEX file. The HEX file is the file that is transmitted from PC to the memory of the microcontroller.

The means by which the Hex file is transmitted from PC to the microcontroller is called a programmer. The programmer generally consists of PC software that manages the hex file to be transferred. A cable between the pc and microcontroller must exist, whether serial cable or parallel port or USB. Most of the manufacturers of microcontrollers provide Programmers and compilers.
2.4.5 Applications of Microcontrollers

- Embedded applications:

  Microcontroller is embedded into a system to play the role of microprocessor core and use its built-in peripherals to see and change the system variables. Such embedded systems that microcontroller could be used for are: automobile, PC, air conditioner, video recorder, mobile telephone, TV, or electronic rice cooker, and so on.

- Control Applications:

  In control applications, the microcontroller must be able to play the control role efficiently. Because physical quantities in the natural world are analog variables, the output signals of a sensor are analog signals. ADC is used to convert these signals from analog to digital. Output signals of a microcontroller are digital signals, so DAC is used to interface the microcontroller with the analog world.

![Applications of Microcontrollers](image_url)
CHAPTER 3  METHODOLOGY AND DESIGN

3. METHODOLOGY AND DESIGN

This chapter presents the design and architecture used to come up with our final Building Automation Over Power Line (BAOPL) system.

First the main goals which the system was designed to achieve are introduced. Then a general description of the system requirements followed by the system scenario to produce the final output is given.

Then depending on that scenario; the modular design of the system is deduced and generally described. Then all modules that constitute the system are presented in details.

3.1 Design Goals

3.1.1 User convenience and system portability

The main goal about Building automation systems is to ease the interaction between the user and the system. In building automation over power line project, the unprofessional user can interact directly with the power line (Plug in/out systems to the socket, install/remove components) and the PC (log in/out to system interface, set/change parameters, observe system status). So it must be necessary to take in mind selecting user friendly components and packaging them into portable modules.

3.1.2 Hardware independence

One key factor in designing such a practical system is its ability to perform equally with hardware components from various vendors; this includes desktops/laptops, controllers and interfacing cards.

Therefore; one of our major goals was to design hardware-independent systems and open messaging protocol.

3.1.3 System integration

The integration of any system is an important requirement. When system integration is not full, the system will fall down and cannot perform its function in a proper manner. One way to verify the system integration is the modular design technique which divides the entire system into individual modules. Every module could be designed and implemented individually, thus the whole system can be represented as individual blocks. Integration of each block could be easily verified; the integration of whole system also could be easily verified.
3.2 System requirements

3.2.1 Establishment of a Power line communication network:
- Per-installed house hold power line.
- Power line transceivers.

3.2.2 Design of simple illustrative modules:
Design simple system modules based on distributed microcontrollers’ organization.
- Air conditioning system.
- Lighting control system.

3.2.3 Monitor the BAOPL system on PC:
- Convenient Human Machine Interface (HMI).

3.3 System Layout

As mentioned in section 3.2, the designed system is consisted of:
- Power line network.
- HMI programmed in PC.
- Use PLM which interfaces the HMI to the Power line.
- Control modules including their I/O devices and PLMs.

The HMI represents the user interface, the control modules represent the control circuits, and the power line network represents the control data high way. The HMI and control modules are connected to the power line network via their PLMs using the same messaging protocol. The system layout is illustrated in figure3.1.
Table 3.1 Key table of BAOPPL system block diagram

<table>
<thead>
<tr>
<th>Item</th>
<th>Abbreviation meaning</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMI</td>
<td>Human Machine Interface</td>
<td>User interface</td>
</tr>
<tr>
<td>PLM</td>
<td>Power Line Modem</td>
<td>Interfacing modules to the power line network</td>
</tr>
<tr>
<td>TS</td>
<td>Temperature Sensor</td>
<td>Measuring the ambient temperature</td>
</tr>
<tr>
<td>SI</td>
<td>Status Indicator</td>
<td>Showing the system status</td>
</tr>
<tr>
<td>IC</td>
<td>Interface Circuit</td>
<td>Interfacing the control circuit to the high voltage appliance</td>
</tr>
<tr>
<td>A/C</td>
<td>Air Conditioner module</td>
<td>The appliance itself</td>
</tr>
<tr>
<td>MS</td>
<td>Motion Sensor</td>
<td>Counting the people inside the place</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro-Controller Unit</td>
<td>The controller module</td>
</tr>
</tbody>
</table>

3.4 System scenario

From figure 3.1, the BOAPPL system was divided into three main parts, the Air conditioning system, the lighting control system, and the User interface module. Therefore it is better to talk about them individually.
3.4.1 Air conditioning system:

The air conditioning system consists of two modules, the conditioner module (air conditioner itself) and the control module. The conditioner module has adjustable parameters, like the ventilation level (speed of motor High/Low) and the cooling level (cool/vent), these parameters are changed by the control module. The core of control module is proposed to be a microcontroller (MCU), the MCU controls the parameters of conditioner unit either manually by explicit user command or automatically by using the temperature sensor output and comparing them with the saved ranges.

3.4.2 Lighting control system:

The lighting control system consists of two modules, lighting module (lamps themselves) and the control module. Normally, the controller module checks the system parameters periodically while observing the network traffic, if the controller received a message from data the user command, The control module changes the system parameters (Turn ON/OFF) either manually by explicit user command or automatically (counting number of present people and turn off the system when there is no body inside the place).

3.4.3 User Interface module:

The User interface module is divided into two parts, HMI and user PLM. The HMI monitors the BAOPL system in a user friendly form. From the HMI, the user can issues commands or observe the system status. Firstly the user plugs his PC (which containing the HMI and connected to the user PLM) to the power line (control network), and then starts the HMI application software. Normally the HMI monitors the system status periodically.

When the user module sent a message dedicated to a certain system, the intended controller module will stop his work to translate and serve the user request and then return back to its normal operation state.

3.5 Development Tools

3.5.1 Hardware tools:

As shown in figure3.1, the BAOPL system was based on the distributed MCUs architecture; therefore the hardware design has concerned to integrate the system controller modules with their peripherals.
Firstly, the specifications of all components were determined to build the design according to those specifications.

Table 3.2 The main Hardware components

<table>
<thead>
<tr>
<th>Component description</th>
<th>Technical description</th>
<th>No of required units</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC Programmer</td>
<td>MicroPro-USB PIC programmer</td>
<td>1</td>
</tr>
</tbody>
</table>

The User interface module

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>HMI</td>
<td>Desktop/Laptop</td>
<td>1</td>
</tr>
<tr>
<td>PLM</td>
<td>PLC_UART/RS232</td>
<td>1</td>
</tr>
</tbody>
</table>

The Lighting control system

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>PIC16F877</td>
<td>1</td>
</tr>
<tr>
<td>Current amplifier</td>
<td>ULN2003</td>
<td>1</td>
</tr>
<tr>
<td>Photo resistor</td>
<td>LDR</td>
<td>1</td>
</tr>
<tr>
<td>Motion detector</td>
<td>IR detectors (Tx/Rx)</td>
<td>2</td>
</tr>
<tr>
<td>Lamp</td>
<td>AC Lamp</td>
<td>1</td>
</tr>
</tbody>
</table>

The Air conditioning system

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCU</td>
<td>PIC16F877</td>
<td>1</td>
</tr>
<tr>
<td>Current amplifier</td>
<td>ULN2003</td>
<td>1</td>
</tr>
<tr>
<td>Temperature sensor</td>
<td>LM35 “temperature sensor”.</td>
<td>1</td>
</tr>
<tr>
<td>Air Conditioner</td>
<td>PC Fan + 12VDC motor.</td>
<td>1</td>
</tr>
</tbody>
</table>

3.5.2 Software Tools

3.5.2.1 MicroSoft Visual Basic 6 (MS VB 6)

From the system scenario (section 3.4.3), obviously, the HMI interacts with the user directly, so the event-oriented programming technique is the most suitable one.

Why was MS Visual Basic selected?

- Visual Basic is a tool that allows developing Windows (Graphic User Interface) applications. The applications have a familiar appearance to the user.[10]
Visual Basic is event-driven, meaning code remains idle until called upon to respond to some event (button pressing, mouse clicking, and so on). Once an event is detected, the code corresponding to that event (event procedure) is executed. [10].

![Event Handling in Visual Basic](image)

**Figure 3.2 Event handling in Visual Basic**

General features of MS VB 6: [10]
- Full set of objects, icons, and pictures.
- Responses to mouse and keyboard actions.
- Direct access to Clipboard and printer.
- Can handle fixed and dynamic variable and control arrays.
- Useful debugger and error-handling facilities.
- Powerful database access tools.

3.5.2.2 CCS compiler software

This compiler was specially designed to meet the special needs of the PICmicro® MCUs. Its tools allow developers to quickly design application software for PIC MCUs in C language.[11]

3.5.2.3 MicroPro-USB PIC programmer software

The MicroPro-USB programmer software, is an application software working under Windows operating system. This software manages the data transfer between the PC (source of HEX file) and programmer hardware to burn the HEX code into the MCU.

This software is available in website www.kitsrus.com.

3.5.2.4 Proteus 7.1

Proteus 7.1 is one of best electronic circuits’ simulators; it was used to simulate the hardware design to insure their performance before starting the implementation.
3.6 Design of System components

3.6.1 Power Line Modem (PLM)

In section 2.3 of chapter 2, the theoretical design requirements of the PLM were mentioned. In this section the practical aspects of PLM design section were discussed.

Generally implementation of a PLM needs high manufacturing technologies (which are not available in developing countries like Sudan), according to this reason, the efforts were oriented to search about manufactured PLM but with suitable specifications. Those specifications were:

- Dedicated to communicate over 220V/50Hz AC power line.
- Uses high performance modulation technique (FSK, PSK, an so on). [see section 2.2.6.3]
- Acceptable data rate (not less than 10Kbps).[1]
- Acceptable transmission range (not less than 20 meters).
- Has USART communication port (general MCU communication port).
- Has RS-232 or USB communication port (general PC ports).

The above specifications were being taken in mind during the searching on web. After about two weeks of continuous searching, a PLM exactly satisfies those specification has been found. This PLM is called “PLC-UART/RS232” from “LinkSprite Technologies, Inc”. [12]

3.6.2 PLC-UART/RS232 Power line modem

PLC-UART/RS232 is a transceiver module designed by “LinkSprite Technologies, Inc” to transparently move serial data over the power line network, and achieves the target of replacing cables by the ubiquitous power line network.

PLC-UART/RS232 has a built-in packet-level repeater function; this feature can greatly extend the coverage of the power line communication. Also it has both physical and logic addresses. In a network, both physical and logic addresses can be used to address different nodes in the network.[12].
Table 3.3 PLC-UART/RS232 Specifications [12]

<table>
<thead>
<tr>
<th>Interface</th>
<th>3.3V TTL UART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optional interface cards:</td>
<td></td>
</tr>
<tr>
<td>▪ RS232 (UART-RS232)</td>
<td></td>
</tr>
<tr>
<td>▪ RS485 (UART-RS485)</td>
<td></td>
</tr>
<tr>
<td>▪ USB (UART-USB)</td>
<td></td>
</tr>
<tr>
<td>▪ Ethernet (UART-Ethernet)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Communication Line Voltage</th>
<th>220VAC/50Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>110VAC/60Hz</td>
<td></td>
</tr>
<tr>
<td>0-400V DC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supply Voltages</th>
<th>external 12V DC supply</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Modulation</th>
<th>FSK</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Carrier frequency</th>
<th>262K/144KHz</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Error Correction</th>
<th>FEC (Forward Error Correction)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Data rate on power line</th>
<th>30Kbps</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Maximum packet data length</th>
<th>300bytes</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Transmission distance</th>
<th>300 ft /91m (without repeater)</th>
</tr>
</thead>
</table>

Figure 3.3 Functional Diagram of PLC-UART/RS232 [12]
3.6.3 RS-232 Communication protocol

RS-232 (Recommended Standard 232) is a standard for serial binary data signals connecting devices. It is commonly used in computer serial ports. Both synchronous and asynchronous transmissions are supported by the standard.

- DTE & DCE concept:

  The RS-232 standard defines the communicating devices as either DTE (Data Terminal Equipment) or DCE (Data Communication Equipment). A typical DTE Device is a computer and a typical DCE Device is a Modem. Thus a DTE to DCE communication is used.

- RS-232 frame format:
  - Start bit.
  - Data (5 to 9 bits of data including parity bit).
  - Parity {odd, even or none}.
  - Stop bits {1, 2}.

![Figure 3.4 RS-232 frame format](image)

The frame format used in this project is: one start bit, 8 data bits, no parity and one stop bit.

3.6.4 USART Module

The USART (Universal Synchronous Asynchronous Receiver Transmitter) {sometimes it is called UART} is the module that performs parallel to serial conversion and vice versa. It can implement the RS232 requirement for data frames and baud rates. The following settings were used to achieve communication.

- Baud rate: 9600 bps.
- Frame: 1 start bit, 8 data bits, no parity and one stop bit.
- Enable both transmitter and receiver.
- Asynchronous transmission.
- Enable interrupt on reception.
3.6.5 BAOPL Protocol

BAOPL protocol is a private protocol designed to match the BAOPL system. It is an application layer protocol follows the manner of master/slave handshaking. Also it identifies each node connected to the power line network using its private address.

BAOPL is only software protocol, so it was embedded into the HMI and the other two sub-systems. The frame format of BAOPL protocol looks like MODBUS protocol, in other words, it could be considered mini-MODBUS protocol.

3.6.5.1 Message frame format:

In BAOPL system, PC interface uses the RS-232 communication port while the MCUs use the UART communication port. Generally the frame formats of UART and RS232 are same, and the data is sent in a string of ASCII characters. BAOPL protocol utilizes that data frame and embeds its frame inside that string.

BAOPL data frame consists of:

1. Private address; identifies the source/destination of messages.
2. Function code; specifies function to be done or response to be received.
3. Data code; holds the value of function/response parameter.

3.6.5.2 Message transaction approach:

HMI is the master controller in BAOPL, while MCUs are the slave controllers. The master controller requests services from the slave controller. The slave controllers are listening to master requests. Once the HMI decided to request a service, it initiates the request prior by address of the intended MCU, say MCU.A, MCU.A will instantly respond to HMI request (check the identification address), decode both of Function and data codes, serve the request, and finally send a notification response to the HMI. During this scenario, the rest of MCUs neither are responding nor interfering. This approach is called the polling approach.
3.6.6 Air conditioner system

3.6.6.1 System layout

The air conditioning system is one of the main BAS items. In BAOPL system, an illustrative model of air conditioning system was designed, this system contains:

- The controller module: a microcontroller was selected to play this role.
- The I/O devices:

  Air conditioning system has three I/O devices; a temperature sensor to measure the ambient temperature (Input), control port interfaces the control module with the conditioner module, and a group of indication LEDs for local indication.

- The communication port: UART communication port was proposed to link the controller module with its PLM.
• PLM: the PLM accomplishes the data exchange process and links the system to the power line network.

• The conditioner module: is an illustrative model of air conditioner, contains simulative components.

• Coupling circuit: couples the low current controller module to the high current conditioner module.

3.6.6.2 The controller module:

The controller module depends mainly on the MCU.

- The list of MCU tasks:

  The MCU is the control unit of the control module; where all control decisions are taken. The main tasks of MCU are listed below:

  ▪ Manages the local control operations.
  ▪ Receives the control commands of HMI.
  ▪ Store all settings into its EEPROM.
  ▪ Receives the temperature reading from the TS (periodically) and uses it in control operation.
Sends feedback to the HMI (system status, set points).

The microcontroller selected for the design is PIC16F877. It includes all the peripherals needed to achieve the above mentioned tasks. Those peripherals are namely timer, USART, EEPROM and ADC.

- The MCU firmware:

  The firmware is the program burned into the memory of the microcontroller. The firmware regulates the performance of the hardware modules in the chip. Every module contains many registers whose values determine the performance of that particular module. The main tasks of the firmware are listed below:
  
  - Initialize the USART to the mentioned settings.
  - Load set point values and last system status from EEPROM.
  - Decode the loaded values and initiate the control parameters.
  - Periodically measure of temperature value (timer0 overflow) via ADC module.
  - Respond and serve the HMI requests.

- The Firmware algorithm:

  1. Load the set points and last system status:
     - Read EEPROM bytes (0→3) and save them into setting matrix.
     - Read EEPROM byte (8), decode its contents, and save it into the system parameters.
  2. Initialize the control parameters
     - Initialize the variables.
     - Run the control procedures.
  3. Initialize the USART module:
     - Set the baud rate to 9600 bps.
     - Enable transmitter and receiver.
     - Set the frame format to (8 data bits, no parity, and one stop bit).
     - Enable receiver interrupt.
     - Set communication mode to asynchronous.
  4. Enable timer0:
     - Set timer0 to approximately one second.
     - Enable timer0 interrupt.
  5. At this point, the program goes into an infinite loop waiting for an interrupt to occur.
• USART Receive Interrupt Service Routine (RISR):

  This interrupt occurs when a new message received.

  The steps:

  ▪ Use the first byte of message to check the address.
  ▪ If the address is F1H then decode the message, else ignore this message and Return back to the infinite loop.
  ▪ Message Decoding steps
     1. Extract the function code from the 2\textsuperscript{nd} byte (b\textsubscript{7}, b\textsubscript{6}).
     2. Redirect the rest of the message to be decoded in one of the following procedures:
        ▪ Operation procedure: Extract the control parameters from the 2\textsuperscript{nd} byte (b\textsubscript{5}→b\textsubscript{0}).
        ▪ Setting procedure: Extract the set points of temperature ranges from bytes (B3→B6).
        ▪ Feedback procedure: Compose a feedback message: (MCU address) (System status, set points values) and send it to HMI. Then return back to the infinite loop.

• Timer0 Overflow (OISR):

  ▪ Initiate its counter to zero.
  ▪ Read the ADC input.
  ▪ Calibrate the read value. Temp = \frac{ADC value \times 5}{Sensor resolution \times ADC resolution}
  ▪ Return back to program.
Start

Initialize MCU

Load last system status

Initialize control parameters

Initialize USART module

Initialize Timer0 module

Execute setting procedure

A

Execute operation procedure

B

Interrupt occur?

No

Yes

Go to ISR

Figure 3.7 Firmware of Air Conditioning system: Flow chart of MCU.
Figure 3.8 Firmware of Air Conditioning system: Flow chart of USART ISR
3.6.7 Lighting control system

3.6.7.1 System layout

In BAOP system, an illustrative model of lighting control system was designed, this system contains:

- The controller module: a microcontroller was selected to play this role.
- The I/O devices: Lighting control system has three I/O devices; a photo sensor to detect whether lamp is on or off (Input), control port interfaces the control module with the lighting module, and a group of indication LEDs for local indication.
- The communication port: UART communication port was proposed to link the controller module with its PLM.
- PLM: the PLM accomplishes the data exchange process and links the system to the power line network.
- The lighting module: is an illustrative model of lighting system, contains just one lamp.
- Coupling circuit: couples the low current controller module to the high current conditioner module.
3.6.7.2 The controller module:

The controller module depends mainly on the MCU.

- The list of MCU tasks:
  
  The MCU is the control unit of the control module; where all control decisions are taken. The main tasks of MCU are listed below:
  
  - Manage the local control operations.
  - Count the number of present people inside the place.
  - Determine the number of present people.
  - Could automatically open/shutdown the system.
  - Receive the control commands of HMI.
  - Store control parameters into its EEPROM.
  - Observe the traffic of people coming on and out the place.
  - Send feedback to the HMI (system status, number of people).

  The microcontroller selected for the design is PIC16F877. It includes all the peripherals needed to achieve the above mentioned tasks. Those peripherals are namely timer, USART, and EEPROM.

- The MCU firmware:
The main tasks of the firmware are listed below:

- Initialize the USART to the mentioned settings.
- Load the last system status from EEPROM.
- Decode the loaded values and initiate the control parameters.
- Simultaneously check the motion sensor and obtain the number of people.
- Respond and serve the HMI requests.

**The Firmware algorithm:**

1. Load the last system status:
   - Read EEPROM byte (8), decodes its contents, and save it into the system parameters.
2. Initialize the control parameters
   - Initialize the variables.
   - Run the control procedures.
3. Initialize the USART module:
   - Set the baud rate to 9600 bps.
   - Enable transmitter and receiver.
   - Set the frame format to (8 data bits, no parity, and one stop bit).
   - Enable receiver interrupt.
   - Set communication mode to asynchronous.
4. At this point, the program goes into an infinite loop waiting for an interrupt to occur.

**USART Receive Interrupt Service Routine (RISR):**

This interrupt occurs when a new message received.

Steps:

- Use the first byte of message to check the address.
- If the address is D1H then decode the message, else ignore this message and return back to program.
- Message Decoding steps
  1. Extract the function code from the 2nd byte (b7, b6).
  2. Redirect the rest of the message to be decoded in one of following procedures:
     - Operation procedure: Extract the control parameters from the 2nd byte (b5→b0).
     - Feedback procedure:
       Compose a feedback message: (MCU address) (System status, number of people) and send it to HMI. Then return back to the infinite loop.
Figure 3.8 Firmware of Lighting control system: Flow chart of MCU.
3.6.8 The HMI Module

3.6.8.1 List of HMI tasks:

The HMI was designed to perform the following tasks:
• Monitor whole of BAOPL system (Air conditioner, Lighting system) in a graphical interface.
• Enable user to set and change the set points and control parameters for each sub-system individually.
• Manage the process of network traffic (Master of network partners).

As mentioned previously (in section 3.5.2.1), Visual Basic programming language is proposed to be used in forming the HMI.

3.6.8.2 HMI Firmware:
• The firmware algorithm:
  1. Load the visual interface.
     ▪ Initialize the parameters and objects.
     ▪ Load the project form.
  2. Configure the communication
     ▪ Set COM1 properties (9600,8,N,1)
     ▪ Open port COM1.
     ▪ Enable Receive interrupt.
  3. Initialize the timer1
     ▪ Set interval length = 100 ms
     ▪ Enable timer1.
  4. Enter into infinite loop
     ▪ Pass the control to the Event processor.
     ▪ Wait an interrupt to occur, then pass control to the interrupt ISR.

• COM1 receive ISR algorithm:
  1. Investigate the address of sender.
  2. Decode the feedback message.
  3. Fill the feedback window of sender by the received values.
  4. Return the control to the Event Processor.

• Timer1 ISR:
  1. Initiate the timer to 0.
  2. Find address of the least Recently Updated sub-system.
  3. Initiate a feedback request relative to that sub-system.
4. Send the feedback request.
5. Return the control to the Event Processor.

- User Command ISR:
  1. Compose the command message
     - Form the destination address.
     - Form the function code.
     - Form the value of function.
  2. Prepare the channel
     - Disable Timer1.
     - Wait for 100 ms.
     - Send the user command.
     - Update the user window.
     - Return control to the Event Processor.
Figure 3.10 HMI module: Firmware flow chart
Figure 3.11 HMI module: Flow chart of COM1 Receive ISR

- Decompose the message
- Sender Address
- Valid
- Extract the Feedback message
- Not valid
- Refill the feedback window

Figure 3.12 HMI module: Flow chart of COM1 Receive ISR

- Timer1 ISR
- Initiate Timer1 to 0
- Destination Address = Least Recently Updated
- Initiate the Feedback message
- Send the feedback request
Figure 3.13 HMI module: Flow chart of user ISR
4. IMPLEMENTATION AND RESULTS

4.1 Overview

This chapter is a continuity of the last chapter (Methodology and Design). This chapter describes the implementation steps, used components, done tests and obtained results. The HMI delivers data to the PLM to inject it into the power line network. Each subsystem has its own PLM, so they receive these data and then deliver it to their controllers to deal with.

4.2 Implementation Plan

The implementation plan is concluded in the following steps:

1. Translate the firmware algorithm programs by using the appropriate software tool.
   - Use “MS Visual Basic 6.0” to write and debug the visual interface.
   - Use “CCS compiler” to write, debug and generate the HEX files.

2. Soft simulation:
   - Use “Proteus 7.1” to construct simulation circuits for individual sub-systems.
   - Insert the generated HEX files in the simulation.
   - Run the simulation and observe the performance in various conditions.
   - Finally deduce and correct design errors.

3. Hardware implementation:
   - Draw the circuit diagrams.
   - Build the circuits step by step.
   - Test modules individually.
   - Deduce and correct the errors.
   - Integrate the system modules.
   - Test performance of entire system.

4. Final tests:
   - Verify the connection over the power line.
   - Connect sub-systems individually and verify their functions.
   - Connect the HMI software and verify its function.
   - Investigate the system integrity.
   - Deduce and correct errors.
### 4.3 Implementation tools and components

The implementation components are summarized in the following table:

<table>
<thead>
<tr>
<th>Module</th>
<th>Notation</th>
<th>Name</th>
<th>Functional description</th>
<th>Technical description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HMI module</strong></td>
<td>HMI</td>
<td>Graphical User Interface Controller</td>
<td>User Graphical Interface</td>
<td>Desktop or Laptop</td>
</tr>
<tr>
<td>PLM1</td>
<td>Power Line Modem No 1</td>
<td>Power line transceiver</td>
<td>PLC_RS232</td>
<td></td>
</tr>
<tr>
<td><strong>Lighting control system</strong></td>
<td>PLM2</td>
<td>Power Line Modem No 2</td>
<td>Power line transceiver</td>
<td>PLC_USART</td>
</tr>
<tr>
<td>MCU1</td>
<td>Microcontroller</td>
<td>Controller module</td>
<td>PIC16F877</td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>Photo Sensor</td>
<td>To detect the surrounding sighting clearness</td>
<td>LDR</td>
<td></td>
</tr>
<tr>
<td>MS</td>
<td>IR motion sensor</td>
<td>Counting people.</td>
<td>IR detector</td>
<td></td>
</tr>
<tr>
<td>IC1</td>
<td>Interface Circuit No 1</td>
<td>Interfacing between MCU and AC power</td>
<td>Relay + ULN2003</td>
<td></td>
</tr>
<tr>
<td>Lamp</td>
<td>lamp</td>
<td>Lighting appliance</td>
<td>lamp</td>
<td></td>
</tr>
<tr>
<td><strong>Air Conditioning system</strong></td>
<td>PLM3</td>
<td>Power Line Modem No 3</td>
<td>Power line transceiver</td>
<td>PLC_USART</td>
</tr>
<tr>
<td>MCU2</td>
<td>Microcontroller</td>
<td>As a control unit of Air conditioning</td>
<td>PIC16F877</td>
<td></td>
</tr>
<tr>
<td>TS</td>
<td>Temperature Sensor</td>
<td>Measuring temperature</td>
<td>LM35</td>
<td></td>
</tr>
<tr>
<td>IC2</td>
<td>Interface circuit No 2</td>
<td>Interfacing between MCU and AC power</td>
<td>3 relays + ULN2003</td>
<td></td>
</tr>
<tr>
<td>A/C</td>
<td>Conditioner module</td>
<td>Air Conditioner</td>
<td>Illustrative model</td>
<td></td>
</tr>
</tbody>
</table>
4.4 Implementation of HMI

4.4.1 HMI layout

HMI was built by using visual basic 6.0 to achieve all tasks which were stated in chapter three. A snapshot of HMI is shown in Figure 4.1.

![Figure 4.1 Snapshot of HMI interface](image)

4.4.2 Description of HMI

As it is shown in figure 4.1, HMI consists of two major frames and some other buttons:

1. Air conditioner control system frame:

   This frame provides access to the air conditioning system, it contains the following windows:

   I. Operation window: contains all valid operation modes. The user chooses the mode of operation (Automatic, Manual). If the user chooses the manual mode, the operating option has to be chosen. Then he clicks on (Apply) button. Once Apply button is pressed the user command is sent to the air conditioning system.
II. Setting window: Enables user to change the set points of temperature ranges. The user clicks on (Change Limits) button, then type the new ranges into text boxes. By clicking on (Execute Changes) button, the new set points will be sent to the air conditioning system.

III. Feedback window: In this window, the status of the air conditioning subsystem is displayed. The temperature degree, the current operating option and the mode of operation (Automatic, Manual) are the data to be displayed.

IV. Lock/Unlock button: used to lock/unlock the user accessibility to system.

2. Lighting control system frame:
   
   This frame provides access to the lighting control system, it contains the following windows:

   I. Operation window: contains all valid operation modes. The user selects the operation mode (Automatic, Manual), if the user selects the manual mode, he also has to select its option (on/off), then he clicks on (Apply) button..

   II. Feedback window: status of lighting control system is displayed here. The current status of the lamp, number of people in the room and the mode of operation (Automatic, Manual) are the data to be displayed.

   III. Lock/Unlock button: used to lock/unlock the user accessibility to system.

3. “Exit” button: clicking on this button terminates the HMI connection and close the interface.

The full visual basic code is presented in appendix B.

4.5 Implementation of the lighting subsystem

4.5.1 MCU programming development:

The firmware of the MCU described in the design chapter was translated into C programs, these programs were written in “CCS compiler” environment, then converted to HEX file format.

The generated file was firstly embedded into the simulator “Protues7.1” to be verified. After many debugging operations, the performance of the MCU was ideally verified [see appendix A figureA.1].
CHAPTER 4  IMPLEMENTATION AND RESULTS

Then the generated HEX file was burnt into the MCU hardware by using the MicroPro PIC programmer.

4.5.2 **Hardware development:**

As mentioned in the prior chapters, the lighting control system is constituted of many electronic components; the following section presents a brief overview of those components.

4.5.2.1 Brief description of required hardware components:

1. **The Microcontroller:** It receives instructions from the HMI, decodes, understands, and responds to them according to a predetermined procedure. The used MCU was PIC_16F877 which is manufactured by Microchip®. This MCU satisfies all required specifications. Figure 4.2 shows the pinout of PIC16F877 microcontroller and Table.1 shows the specifications of it.

![Figure 4.2 PIC16F877 pinout][13]
Table 4.2 PIC 16F877 specifications [13]

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Frequency</td>
<td>DC - 20 MHz</td>
</tr>
<tr>
<td>FLASH Program Memory</td>
<td>8K (14-bit words)</td>
</tr>
<tr>
<td>Data Memory (bytes)</td>
<td>368</td>
</tr>
<tr>
<td>EEPROM Data Memory</td>
<td>256</td>
</tr>
<tr>
<td>Interrupts</td>
<td>14</td>
</tr>
<tr>
<td>I/O Ports</td>
<td>Ports A,B,C,D,E</td>
</tr>
<tr>
<td>Timers</td>
<td>3</td>
</tr>
<tr>
<td>Serial Communications</td>
<td>MSSP, USART</td>
</tr>
<tr>
<td>Parallel Communication</td>
<td>PSP</td>
</tr>
<tr>
<td>10-bit ADC Modules</td>
<td>8 input channels</td>
</tr>
<tr>
<td>Analog Comparators</td>
<td>2</td>
</tr>
<tr>
<td>Instruction Set</td>
<td>35 Instructions</td>
</tr>
</tbody>
</table>

2. Power line modem:

PLC-UART/RS232 PLM was selected for fulfillment of BAOPL project. The PLC-UART/RS232 is a power line transceiver module from Link Sprite Technologies, Inc. It has satisfactory specifications listed in Table 4.3. Snapshots of the PLM are shown in figure 4.3.

The PLM that is connected to PC is a CDE device, so it could be directly connected to the PC (PC-male connection, PLM-female connection).
Table 4.3: Specifications of PLC-UART/RS232 module [12].

<table>
<thead>
<tr>
<th>Product name</th>
<th>PLC-UART/RS232/RS485 Transceiver Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>UART, RS232 or RS485</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>230VAC/50Hz, 110VAC/60Hz</td>
</tr>
<tr>
<td>Modulation</td>
<td>FSK (Frequency Shift Keying)</td>
</tr>
<tr>
<td>Carrier frequency</td>
<td>262K/144KHz</td>
</tr>
<tr>
<td>Error Correction</td>
<td>FEC (Forward Error Correction)</td>
</tr>
<tr>
<td>Data rate on Powerline</td>
<td>30Kbps</td>
</tr>
<tr>
<td>Maximum packet data length</td>
<td>320 bytes</td>
</tr>
<tr>
<td>Repeater Hops</td>
<td>3 Hops</td>
</tr>
<tr>
<td>Transmission distance</td>
<td>300 feet (no repeater)</td>
</tr>
<tr>
<td>Support nodes number</td>
<td>65535</td>
</tr>
<tr>
<td>LED</td>
<td>Power Line Activity LED system LED serial port LED</td>
</tr>
</tbody>
</table>
3. Photo resistor:

   It is used by MCU to check status of the lamp. Photo resistor (LDR) is a photosensitive sensor; its electric resistance is influenced reversely to the light intensity. Figure 4.4 shows LDR appearance.
4. Motion sensors

Two IR sensors were utilized to sense the entering and leaving bodies. Each IR sensor consists of two parts. IR Emitter emits IR signals with 35KHz square wave with the aid of 555 timer. On the other hand IR Receiver detects the IR signal.

When some body cuts the IR signal the receiver senses no IR signal and changes its “OUT” pin status which is detected by MCU.

5. LM555 single timer:

It is an IC used to produce square signal with certain frequency to be used for triggering the IR emitter (e.g 35KHz). Figure 4.6 shows LM555 connection.
6. ULN2003:

ULN2003 is a type of current amplifier ICs. This IC can interface two different electronic circuits; in this system it protects the MCU (low current element) from the controlled devices (higher current element).

ULN2003 consists of a collection of Darlington gates, the inputs of these gates are connected to the outputs of MCU, while its outputs are connected to the input of the controlled device (such as: relays, dc motors, and so on). ULN2003 pinout is shown in figure 4.7.
4.5.2.2 Hardware connection:

The hardware connection is dependant mainly on the connection of MCU (PIC16F877) to its peripherals.

In the PIC16F877, PIN_C6 and PIN_C7 were connected to the UART interface of the PLM via the communication port, the control signal was fed to the ULN2003 which is connected to a relay via control port. For purpose of local indication a LED was connected to the control port. Also an oscillator of frequency 4683594Hz was connected to the MCU clock input.

The above connection scenario was firstly implemented on test boards and then the function of the system was verified. To insure high reliability and performance, the lighting control module was welded on a printed board as shown in figure 4.8.

4.5.2.3 System packaging:

This control module in addition to the PLM, relay, motion sensors, lamp and the power connections devices were packaged together in one model as shown in figure 4.9.
CHAPTER 4  IMPLEMENTATION AND RESULTS

Figure 4.8 Lighting control system after implementation

Figure 4.9 The packaged model of lighting control system
4.6 Implementation of the Air conditioning system

4.6.1 MCU programming development:

The firmware of the MCU described in the design chapter was translated into C programs, these programs were written in “CCS compiler” environment, then converted to HEX file format.

The generated file was firstly embedded into the simulator “Protues7.1” to be verified. After many debugging operations, the performance of the MCU was ideally verified [see appendix A figureA.1].

Then the generated HEX file was burnt into the MCU hardware by using the MicroPro PIC programmer.

4.6.2 Hardware development:

4.6.2.1 Brief description of required hardware components:

Some of hardware components which were used in the lighting control system were also used in this system; therefore there is no need to describe their features again.

1. Microcontroller: PIC 16F877. (see section 4.5.2.1)
2. Power Line Modem. (see section 4.5.2.1)
3. ULN2003. (see section 4.5.2.1)
4. Temperature sensor (LM35): This device is used for measuring the temperature. The LM35 electric resistance is influenced positively to temperature and it represents that influence in output voltage with a resolution of 0.01V per 1 Celsius degree [15]. This output voltage is an analog signal, so it must go through the ADC module before the digital processing. The pinout of LM35 is shown in Figure4.10.

Figure4.10 LM35 pinout [15]
After the LM35 output signal was converted, some calibration must be done to obtain the real value of temperature. This calibration is done by the MCU by using the following equation.

\[
\text{Temp value} = \frac{\text{ADC reading} \times 5}{100 \times \text{ADC resolution}}
\]

4.6.2.2 Hardware connection:

The hardware connection is dependant mainly on the connection of MCU (PIC16F877) to its peripherals.

In the PIC16F877, PIN_C6 and PIN_C7 were connected to the UART interface of the PLM via the communication port, the control signal was fed to the ULN2003 which is connected to a relay via control port. For purpose of local indication group of LEDs was connected to the control port. Also an oscillator of frequency 4683594Hz was connected to the MCU clock input.

The above connection scenario was firstly implemented on test boards and then the function of the system was verified. To insure high reliability and performance, the lighting control module was welded on a printed board as shown in figure 4.11.

4.6.2.3 System packaging

This control module in addition to the PLM, two relays, PC fan, mini DC motor and the power connections devices were packaged together in one model as shown in figure 4.12.
Figure 4.11 The Air conditioner control module after implementation

Figure 4.12 The packaged model of Air conditioning system
4.7 The results:

After a lot of test operations, calibrations and modifications; the following results were obtained

4.7.1 The HMI implementation results:
   1. HMI Performance:
      After a lot of tests and observations, high efficiency interaction was obtained at 300ms interval length of Timer1; both of optimum response and performance were obtained, therefore the optimum HMI refresh rate was 2 times per second which is perfect for BAS systems.
   2. Validity of BAOPL protocol:
      The BAOPL protocol which had been mentioned in the chapter of design was used to manage the traffic of BAOPL network, this protocol had been built in an extendable form; therefore the integrity of BAOPL system was perfectly verified by using the BAOPL protocol.
   3. Response to user requests:
      Generally, few milliseconds are spent between sending of request and receiving of subsystem response, but the BAOPL network was based on broadcasting topology; therefore it had been needed to prepare channels before starting the data exchange, thus additional few portions of second were spent. As a result and after some attempts, the maximum spent time of user request completion was 0.5 sec.

4.7.2 The sub-systems:
   1. The MCU performance:
      The performance of the MCU was verified by changing the HMI refresh rate; at refresh rate more than 2 times per second, the MCU was being led to a deadlock state and then system restart was required. Therefore the optimum MCU performance was obtained at user request rate not more than 2 times per second.
   2. Motion detection performance:
      The computation of people counting had been embedded in the MCU program and its performance was dependant on the MCU speed, therefore the performance of motion detectors was depending mainly on speed of MCU; the small firmware of Lighting MCU plus the high frequency clock signal led to verification of good counting performance. Another factor of motion detection was the sensitivity of IR motion detectors, experimentally, the sensitivity of motion detectors were obtained as follows: the IR motion detectors could detect any nontransparent body which its thickness is not less than 1cm.
   3. Temperature measuring:
      The accuracy verification of temperature measuring was dependant on two factors, the first one was the accuracy of sensor itself, the second one was the accuracy of ADC module. Determination of sensor accuracy was taken form the data sheet, the accuracy was about 0.5 degree Celsius.
For ADC accuracy, characteristics of ADC conversion were obtained experimentally; the optimum temperature measuring was obtained at ADC resolution of 1024 and conversion rate of one time per 5 sec.
5. CONCLUSION AND FUTURE DEVELOPMENTS

5.1 Conclusion

The project of Building Automation over power line communication was successfully designed and implemented. It is an economical and feasible solution for the proprietary connections which are used in Building Automation Systems by using the household power line for transmitting control data in addition to the AC power.

After comparing the proposed design goals with those accomplished, the main goals are seen to be met.

- The project is a proof of possibility of using the ubiquitous power line as control network.
- No additional efforts or costs are required for installing a new building automation network while the power line can connect among the system subscribers.
- The designed system collects between the properties of hardwired systems and wireless systems; it has a hardwired networked and also has the property of system flexibility.
- The communication cards can be embedded inside the controlled device itself; therefore this device could be plugged in the network any where any time.
- The user can easily access the system from any electrical socket without any proprietary restriction.
- The system is based on the architecture of distributed microcontrollers, this architecture verifies high performance and also it is economically feasible.
- A special protocol was designed and named as “BAOPL” protocol, this protocol looks like the MODBUS protocol which is widely used in control system.
- BAOPL protocol is managing the network traffic perfectly and it was designed in an extendable way, so it possible to use this protocol in huge buildings.
5.2 Problems and solutions

The problems which were encountered in the implementation are listed below:

5.2.1 Broadcasting communication:

- Problem:

  The PLC-RS232/UART power line transceivers are working like broadcasting transceivers, so if any modem sent data this data could be picked up by others modem in the same domain. The BAOPL system presents single domain network; therefore the broadcasting method leads to a problem of no way to fulfill the multi-node networking.

- Solution:

  The PLC-RS232/UART power line transceivers use the serial protocol in the communication, the serial protocol is an open protocol which sends and receives strings of bytes. The first byte of this string was used to hold a private address; this address indicates the sender once a message is received and the receiver when the message is sent.

5.2.2 Single phase restriction:

- Problem:

  The used power line modems were designed to use single phase power lines, this problem has appeared when three-phase power line was being used during the tests in the Laboratory of control, the laboratory has many sockets of three phase; the HMI could not interact with such a device that was connected to another phase.

- Solution:

  The PLC-RS232/UART modem has tools of AT commands, one of those tools is the searching tool; by using this tool all of peer PLMs are shown, if such a PLM was not found, so it had been connected to a different phase and it must be plugged to another phase. The above scenario was simply implemented by a software program while the person was attempting the new connection.
5.3 Future developments:

5.3.1 Outdoor accessibility:

Outdoor accessibility means the possibility to access the BAOP network form outdoor; by using the public electricity network to transmit data through, then the system could be accessed anywhere outdoor the building. This expansion needs additional study, because the outdoor power line environment is noisier than indoor.

5.3.2 Building Management Over Power Line:

The building management concept is more inclusive than the building automation. The building automation is exclusive for the household appliances while the building management including other systems like: complex security systems, attendance systems, private networks, and so on.

Because of power line sockets are distributed anywhere inside the buildings; therefore it is a feasible idea to use the ubiquitous power line in the Building Management Systems.

5.3.3 Other applications of power line communication:

- Outdoor remote access for BAS.
- Streets’ lighting control systems.
- Industrial fields applications.
- Processes of Electric power.
References


Appendix A. Circuit diagrams

Figure A.1 simulation circuit for the lighting control system
Figure A.2 Simulation circuit for air conditioner subsystem
APPENDIX B: PROJECT CODES

B.1 The C language code of the Lighting control MCU:

```
#include <16F877.h>
device adc=8

#define NOWDT   //No Watch Dog Timer
#define HS      //High speed Osc (> 4mhz)
#define NOPUT   //No Power Up Timer
#define PROTECT //Code protected from reads
#define NOBROWNOUT //No brownout reset
#define NOLVP //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#define NOCPD   //No EE protection
#define NOWRT   //Program memory not write protected
#define NODEBUG //No Debug mode for ICD

delay(clock=4683594)
rs232(baud=9600, parity=N, xmit=P8N6 C6, rcv=P8N7 C7, bits=8)

people= the number of present people [0].
state1_past=the past state of detector 1 [0].
state2_past=the past state of detector 2 [0].
state1_present=the present state of detector 1 [0].
state2_present=the present state of detector 2 [0].
state1=counter of state 1 [0].
state2=counter of state 2 [0].
counter1=counter of detector 1 [0].
counter2=counter of detector 2 [0].
cmd=command (mode:option) [0:0].
response=the final system response (OFF/ON) [0].
data= the control data comming from user.

int income, device=0, people=0, state1_past=0, state2_past=0, state1=0,
mode=0, value=0, state2=0,
light_data=0, counter1=0, counter2=0, op_cmd=0, response=0, sensor;

void checkstates(void)
{
    int state1_present=0, state2_present=0, j=0;
    // check detectors
    j=input_d();
sensor = (j>>2)&0x01;
state1_present=j%2;
j=j>>1;
state2_present=j%2;
state1=state1_present+(state1_past*2);   // calculate state1
state2=state2_present+(state2_past*2);   // calculate state2
    // save the current states
    state1_past=state1_present;
    state2_past=state2_present;
}
```
void presencecounter (void)
{
    checkstates();
    if (state1>0 || state2>0) //just determine when there is a
    triggering event
    {
        switch (state1)
        {
            case 1: // state (1,0)
                if ((counter1==0) && (counter2!=2)) {counter1=1; break;}
                if ((counter1==0) && (counter2==2)) {counter2=3; break;}
                break;
            case 2: // state (2,0)
                if ((counter1==1) && (counter2==0)) {counter1=2; break;}
                if ((counter1==0) && (counter2==3)) {counter2=4; break;}
                break;
        }
        switch (state2)
        {
            case 1: // state (0,1)
                if ((counter1==2) && (counter2==0)) {counter1=3; break;}
                if ((counter2==0) && (counter1!=2)) {counter2=1; break;}
            case 2: // state (0,2)
                if ((counter1==3) && (counter2==0)) {counter1=4; break;}
                if ((counter2==1) && (counter1==0)) {counter2=2; break;}
            }
            if (counter1==4) {people++; counter1=0;} // one body entered
            if (counter2==4) {if (people>0) people--; counter2=0;} // one body
            left
        }
    }
}
void feedback_light(void)
{
    value = 0xF1;
    putc(value);
    value = mode+(sensor<<1)+(people<<2);
    putc(value);
}
void decode_light(void)
{
    int j=0;
    device=0;
    j = (light_data>>6);
    switch (j)
    {
        case 0: {
            mode = (light_data>>5)&0x01; write_eeprom(0,mode);
            op_cmd=light_data&0x01; write_eeprom(1,op_cmd);
            break;
        case 1: { break;}
        case 2: { feedback_light(); break;}
    }
default: {break;}
}

// int_RDA

void RDA_isr()
{
  income = getc();
  if ((income>>6)==0x03)
    device=(income>>4)&0x03;
  else
  {
    switch (device)
    {
    case 1: {light_data=income; decode_light(); break;}
    case 2: {break;}
    }
  }

  //// control decision making
  void control (void)
  {
    presencecounter();
    // outcome the number of present people on port A
    switch (mode)
    {
    case 0: {
      if (people==0)
        response=0;
      else
        response=1;
      break;   }
    case 1:{
      response=op_cmd;
      break; }
    }
  output_bit(pin_b0,response); //// observe the control action in portb_0
  output_a(people);
  }

  //// Main program's body
  void main()
  {
    setup_adc_ports(NO_ANALOGS);
    setup_adc(ADC_OFF);
    setup_psp(PSP_DISABLED);
    setup_spi(SPI_SS_DISABLED);
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DISABLED,0,1);
    enable_interrupts(INT_RDA);
    enable_interrupts(GLOBAL);
    
    mode = read_eeprom(0);
    op_cmd = read_eeprom(1);
    while (true)
    {
      control();
    }
B.2 The C language code of the air conditioner MC

```c
#include <16F877.h>
#define adc=10
#define delay(clock=4683594)
#define rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

//---------- Global variables definition -----------------------------
mode: the operation mode.
op_prt: the present state of operation level.
op_cmd : the received operation command.
response_prt: the present state of output.
response_pst: the past state of output.
ctrl : the latest control byte.
counter: general counter used in receiving of data frame.
datalen: the length of incoming data frame.
Tempval: the current temperature value.
fbdata: the feedback contents.
limits Array: array of limits of different levels.
msg Array: contains the incoming data frame.

// the EEPROM ////
eeprom locations:
- 0x00 -> 0x03 [limits array].
- 0x08 [mode: op].

// Global variables declarations //
int mode, op_prt, op_cmd, response_prt=0, response_pst, ctrl=0, count=0, datalen, counter=0, tempval=0, fbdata[7];
n int limits[4]={10,20,30,40}, msg[10];
float y=0;

// update the operation parameters /////
void operation_update (void)
{
    ctrl=read_eeprom(8); // recover the latest control status.
    mode=(ctrl>>5)&0x01; // extract the control mode.
    op_cmd=ctrl&0x1F; // extract the operation level.
}

// update settings /////
void setting_update (void)
{
    int i;
    // extract the setting values from it's EEPROM locations /////
    for (i=0;i<4;i++)
    {
        limits[i]=read_eeprom(i);
        limits[i]=msg[i+1];
    }
}

// construct feedback and then send /////
void feedback (void)
{
    int i;
    fbdata[0]=0xF6; // fill the header byte.
    // fill the rest elements //
```
APPENDIX B

PROJECT CODES

```c
fbdata[1]=tempval;
fbdata[2]=(op_prt<<1)+ mode;
for (i=0;i<4;i++)
    fbdata[i+3]=limits[i];
// send the feedback message ////
for (i=0;i<=6;i++)
   putc(fbdata[i]);

//////// decode the received message ///////////
void decode_msg(void)
{
    int i;
    // discover and then execute the intended process /////
    i=msg[0]>>6;
    switch(i)
    {
        // store and update the operation parameters
        case 0: {  ctrl = msg[0];  write_eeprom(8,(msg[0]&0x3F));
                  operation_update(); break;}
        // store and update the setting parameters
        case 1: {
                  for (i=0;i<4;i++)
                      write_eeprom(i,msg[i+1]);
                  setting_update();
                  break;
                }
        // construct and send the feedback message
        case 2: { feedback(); break;}
    }
}

#define RDA
RDA_isr()
{
    int income;
    income=getc();
    // extract the header of the message
    if ((income & 0xF0)==0xF0)
    {
        count=0;
        datalen=income & 0x0F;
    }
    // extract and store the message elements
    else
    {
        msg[count]=income;
        count++;
    }
    // decode the message if it is received completely
    if (count>=datalen)
        decode_msg();
}

/////////// read the ambient temperature ///////////
void read_temperature(void)
{
    int temp;
    temp=read_adc(); // A/D conversion
```
\[
y = \text{temp} \times \frac{500}{1023}; \quad \text{// transform to real temperature value.}
\]
\[
\text{tempval} = \text{abs}(y); \quad \text{// take the absolute value of temperature.}
\]

```

```
void main()
{
    setup_adc_ports(AN0);
    setup_adc(ADC_CLOCK_INTERNAL);
    setup_psp(PSP_DISABLED);
    setup_spi(FALSE);
    setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
    setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
    setup_timer_2(T2_DISABLED,0,1);
    enable_interrupts(INT_RDA);
    enable_interrupts(INT_TIMER1);
    enable_interrupts(GLOBAL);
    // recover the latest setting parameters even if MCU had been powered off.
    setting_update();
    ///////// Read the temperature of the ambient
    // update the operation with the latest state.
    operation_update();
    while(1)
    {
        // real time observation and control
        control();
    }
}
B.3 Visual Basic Code for HMI software:

```vbnet
Dim AC_mode, AC_opcmd, AC_initial, AC_initdata As Integer
Dim Light_mode, Light_opcmd, Light_initial, Light_init_data As Integer
Dim msg As String
Dim pointer As Integer

Private Sub Form_Load()
    With MSComm1
        .CommPort = 1
        .Settings = "9600,N,8,1"
        .DTREnable = True
        .RThreshold = 1
        .SThreshold = 1
        .InputMode = comInputModeText
        .InputLen = 0
        .InBufferCount = 0
        .PortOpen = True
    End With
    cmdAirCondOpY.Enabled = False
    cmdLightOpY.Enabled = False
    AC_initial = 0
    Light_initial = 0
    pointer = 1
    tmrUpdate.Enabled = True
End Sub

Private Sub cmdExit_Click()
    MSComm1.PortOpen = False
End
End Sub

'///////////////// Begin of Air Conditioner Part ///////////////
Private Sub cmdAirCondOpY_Click()
    Call prepare_channels
    Call update_Airconditioner_status
    cmdAirCondOpY.Enabled = False
End Sub

Private Sub cmdAirCondChngLimit_Click()
    Call change_Airconditioner_SetPoints
    cmdAirCondLimitExe.Visible = True
    cmdAirCondChngLimit.Visible = False
End Sub

Private Sub cmdAirCondLimitExe_Click()
    Call prepare_channels
    Call update_Airconditioner_SetPoints
    cmdAirCondLimitExe.Visible = False
End Sub

Private Sub optAirCondOpAuto_Click()
    AC_mode = 0
    AC_opcmd = 1
    cmdAirCondOpY.Enabled = True
End Sub
```
Private Sub optAirCondOpManOff_Click()
    AC_mode = 1
    AC_opcmd = 0
    cmdAirCondOpY.Enabled = True
End Sub

Private Sub optAirCondOpManLV_Click()
    AC_mode = 1
    AC_opcmd = 1
    cmdAirCondOpY.Enabled = True
End Sub

Private Sub optAirCondOpManHV_Click()
    AC_mode = 1
    AC_opcmd = 2
    cmdAirCondOpY.Enabled = True
End Sub

Private Sub optAirCondOpManLC_Click()
    AC_mode = 1
    AC_opcmd = 3
    cmdAirCondOpY.Enabled = True
End Sub

Private Sub optAirCondOpManHC_Click()
    AC_mode = 1
    AC_opcmd = 4
    cmdAirCondOpY.Enabled = True
End Sub

Private Sub optAirCondOpManPmp_Click()
    AC_mode = 1
    AC_opcmd = 5
    cmdAirCondOpY.Enabled = True
End Sub

Private Sub update_Airconditioner_status()
    Dim ACoperation, ACdevice, ACdatalen, ACheader, ACctrl As Integer
    ACdevice = 3
    ACdatalen = 1
    ACoperation = 0
    ACheader = 192 + ACdevice * 16 + ACdatalen
    ACctrl = ACoperation * 64 + AC_mode * 32 + AC_opcmd
    MSComm1.Output = Chr(ACheader) & Chr(ACctrl)
End Sub

Private Sub check_Airconditioner_status()
    Dim ACoperation, ACdevice, ACdatalen, ACheader, ACctrl As Integer
    ACdevice = 3
    ACdatalen = 1
    ACoperation = 2
    ACheader = 192 + ACdevice * 16 + ACdatalen
    ACctrl = ACoperation * 64
    MSComm1.Output = Chr(ACheader) & Chr(ACctrl)
End Sub
Private Sub decode_msg_Airconditioner()
    Dim ACfheader, ACfbmsg, ACfbmode, ACfbop, fbdevice As Integer
    ACfheader = Asc(Left(msg, 1))
    fbdevice = ((ACfheader \ 16) Mod 4)
    If (fbdevice = 3) Then
        lblTemperature.Caption = "Temperature degree:   " & Asc(Mid(msg, 2, 1))
        ACfbmsg = Asc(Mid(msg, 3, 1))
        ACfbmode = ACfbmsg Mod 2
        ACfbop = (ACfbmsg Mod 16) \ 2
        Select Case ACfbmode
            Case 0
                lblMode.Caption = "Mode:  Automatic"
            Case 1
                lblMode.Caption = "Mode:  Manual"
        End Select
        Select Case ACfbop
            Case 0
                lblStatus.Caption = "Status:  OFF"
            Case 1
                lblStatus.Caption = "Status:  Low Vent"
            Case 2
                lblStatus.Caption = "Status:  High Vent"
            Case 3
                lblStatus.Caption = "Status:  Low Cool"
            Case 4
                lblStatus.Caption = "Status:  High Cool"
            Case 5
                lblStatus.Caption = "Status:  Pump Only"
        End Select
        If (cmdAirCondLimitExe.Visible = False) Then
            Call decode_limits
        End If
    End If
End Sub

Private Sub decode_limits()
    txtAirCondLVSP.Visible = False
    txtAirCondHVSP.Visible = False
    txtAirCondLCSP.Visible = False
    txtAirCondHCSP.Visible = False
    lblAirCondLVSP.Caption = "LV:" & Asc(Mid(msg, 4, 1))
    lblAirCondHVSP.Caption = "HV:" & Asc(Mid(msg, 5, 1))
    lblAirCondLCSP.Caption = "LC:" & Asc(Mid(msg, 6, 1))
    lblAirCondHCSP.Caption = "HC:" & Asc(Mid(msg, 7, 1))
    cmdAirCondChngLimit.Visible = True
End Sub
Private Sub change_Airconditioner_SetPoints()
    txtAirCondLVSP.Visible = True
    txtAirCondHVSP.Visible = True
    txtAirCondLCSP.Visible = True
    txtAirCondHCSP.Visible = True
    txtAirCondLVSP.Text = Mid(lblAirCondLVSP.Caption, 4, 3)
    txtAirCondHVSP.Text = Mid(lblAirCondHVSP.Caption, 4, 3)
    txtAirCondLCSP.Text = Mid(lblAirCondLCSP.Caption, 4, 3)
    txtAirCondHCSP.Text = Mid(lblAirCondHCSP.Caption, 4, 3)
End Sub

Private Sub update_Airconditioner_SetPoints()
    Dim ACdevice, ACdatalen, ACoperation, ACheader, ACctrl As Integer
    ACdevice = 3
    ACdatalen = 5
    ACoperation = 1
    ACheader = 192 + ACdevice * 16 + ACdatalen
    ACctrl = ACoperation * 64
    pointer = 3
    MSComm1.Output = Chr(ACheader) & Chr(ACctrl) & Chr(Val(txtAirCondLVSP)) & Chr(Val(txtAirCondHVSP)) & Chr(Val(txtAirCondLCSP)) & Chr(Val(txtAirCondHCSP))
End Sub

Private Sub initialize_AirCond()
    Dim ACfbop, ACfbmode As Integer
    ACfbmode = AC_init_data / 16
    ACfbop = AC_init_data Mod 16
    Select Case ACfbmode
        Case 0
            optAirCondOpAuto.Value = True
        Case 1
            Select Case ACfbop
                Case 0
                    optAirCondOpManOff.Value = True
                Case 1
                    optAirCondOpManLV.Value = True
                Case 2
                    optAirCondOpManHV.Value = True
                Case 3
                    optAirCondOpManLC.Value = True
                Case 4
                    optAirCondOpManHC.Value = True
                Case 5
                    optAirCondOpManPmp.Value = True
            End Select
    End Select
    cmdAirCondOpY.Enabled = False
    AC_initial = 1
End Sub

Private Sub cmdAirconditionerLock_Click()
    framAirCondOp.Enabled = False
    framAirCondSP.Enabled = False
    cmdAirconditionerUnlock.Enabled = True
    cmdAirconditionerLock.Enabled = False
End Sub

Private Sub cmdAirconditionerUnlock_Click()
    framAirCondOp.Enabled = True
    framAirCondSP.Enabled = True
    cmdAirconditionerLock.Enabled = True
    cmdAirconditionerUnlock.Enabled = False
End Sub

Private Sub cmdLightLock_Click()
    framLightOp.Enabled = False
    cmdLightLock.Enabled = False
    cmdLightUnlock.Enabled = True
End Sub

Private Sub cmdLightUnlock_Click()
    framLightOp.Enabled = True
    cmdLightLock.Enabled = True
    cmdLightUnlock.Enabled = False
End Sub

'/////////// End of Air Conditioner Part///////////

'/////////// Begin of Lighting System Part///////////
Private Sub cmdLightOpY_Click()
    Call prepare_channels
    Call update_light_status
    cmdLightOpY.Enabled = False
End Sub

Private Sub optLightOpAuto_Click()
    Light_mode = 0
    Light_opcmd = 1
    cmdLightOpY.Enabled = True
End Sub

Private Sub optLightOpManOff_Click()
    Light_mode = 1
    Light_opcmd = 0
    cmdLightOpY.Enabled = True
End Sub

Private Sub optLightOpManOn_Click()
    Light_mode = 1
    Light_opcmd = 1
    cmdLightOpY.Enabled = True
End Sub

Private Sub check_Light_status()
    Dim Ldevice, Ldatalen, Loperation, Lheader, Lctrl As Integer
    Ldevice = 1
    Ldatalen = 1
    Loperation = 2
    Lheader = 192 + Ldevice * 16 + Ldatalen
    Lctrl = Loperation * 64
    MSComm1.Output = Chr(Lheader) & Chr(Lctrl)
End Sub
Private Sub update_light_status()
    Dim Ldevice, Ldatalen, Loperation, Lheader, Lctrl As Integer
    Ldevice = 1
    Ldatalen = 1
    Loperation = 0
    Lheader = 192 + Ldevice * 16 + Ldatalen
    Lctrl = Light_opcmd + Light_mode * 32 + Loperation * 64
    ' pointer = 1
    MSComm1.Output = Chr(Lheader) & Chr(Lctrl)
End Sub

Private Sub decode_msg_Light()
    Dim Lfbheader, Lfbmsg, Lfbmode, Lfbop, fbpeople As Integer
    Lfbheader = Asc(Left(msg, 1))
    fbdevice = Lfbheader Mod 16
    If (fbdevice = 1) Then
        Lfbmsg = Asc(Mid(msg, 2, 1))
        Lfbmode = Lfbmsg Mod 2
        Lfbop = (Lfbmsg Mod 4) \ 2
        fbpeople = Lfbmsg \ 4
        If (Lfbmode = 0) Then
            lblLightFBMode.Caption = "Mode:    Automatic"
        Else
            lblLightFBMode.Caption = "Mode:    Manual"
        End If
        If (Lfbop = 0) Then
            lblLightFBStatus.Caption = "Status:    OFF"
        Else
            lblLightFBStatus.Caption = "Status:    ON"
        End If
        lblLigtFBPeople.Caption = "Number of present people:  " & Chr(13) & " & fbpeople
    End If
    'If (Light_initial = 0) Then
    '    Light_init_data = Lfbop + (Lfbmode * 16)
    '    Call initialize_Light
    'End If
End Sub

Private Sub initialize_Light()
    Dim Lfbop, Lfbmode As Integer
    Lfbmode = Light_init_data / 16
    Lfbop = Light_init_data Mod 16
    Select Case Lfbmode
        Case 0
            optLightOpAuto.Value = True
        Case 1
            Select Case Lfbop
                Case 0
                    optLightOpManOff.Value = True
                Case 1
                    optLightOpManOn.Value = True
            End Select
    End Select
    cmdLightOpY.Enabled = False
    Light_initial = 1
End Sub  
'----------------------------------- End of Lighting System Part -----------------------------------  

Private Sub prepare_channels()
    Dim ready As Integer
    tmrUpdate.Enabled = False
    ready = 0
    Do While (ready < 50)
        ready = ready + 1
        Debug.Print ready
    Loop
End Sub

Private Sub MSComm1_OnComm()
    If (MSComm1.CommEvent = comEvReceive) Then
        msg = MSComm1.Input
        Text1.Text = msg
        Select Case (Len(msg))
            Case 7:
                Call decode_msg_Airconditioner
            Case 2:
                Call decode_msg_Light
        End Select
    End If
    If (MSComm1.CommEvent = comEvSend) Then
        tmrUpdate.Enabled = True
    End If
End Sub

Private Sub tmrUpdate_Timer()
    ' Select Case (pointer)
    '    Case 1:
    '        Call check_Light_status
    '        pointer = 3
    '    Case 3:
    '        Call check_Airconditioner_status
    '        pointer = 1
    'End Select
End Sub