

Palestine Polytechnic University



**College of Engineering & Technology
Electrical and Computer Department**

Graduation Project

Monitoring Home By Cellular Technology

Project Team

Manar Ikhlayile

Sabha Abu Sabha

Yasmin Al-Shamesti

Project Supervisor
Eng. Ayman Wazwaz

Hebron – Palestine
Dec, 2010

Palestine Polytechnic University



**College of Engineering & Technology
Electrical and Computer Department**

Graduation Project

Monitoring Home By Cellular Technology

Project Team

Manar Ikhlayile

Sabha Abu Sabha

Yasmin Al-Shamesti

Project Supervisor
Eng. Ayman Wazwaz

Hebron – Palestine
Dec, 2010

Monitoring Home By Cellular Technology

Project Team

Manar Ikhlayile Sabha Abu Sabha Yasmin Al-Shamesti

Project Supervisor

Ayman Wazwaz

This Under-Graduate Project Report submitted to Computer and Electrical engineering “Department in College of Engineering and Technology

Palestine Polytechnic University

For accomplishment the requirements of the bachelor degree in Engineering field at Computer System Engineering

**Palestine Polytechnic University
Hebron – Palestine
June, 2010**

جامعة بوليتكنك فلسطين

الخليل – فلسطين

كلية الهندسة و التكنولوجيا

دائرة الهندسة الكهربائية والحاسوب

Monitoring Home By Cellular Technology

صبحة عيسى أبو صبحة منار حسين اخليل ياسمين يوسف الشمسلي

على نظام كلية الهندسة والتكنولوجيا و إشراف المشرف

تقديم هذ إلى دائرة الهندسة الكهربائية و بمتطلبات درجة البكالوريوس في الهندسة
تخصص هندسة أنظمة الحاسوب.

توقيع المشرف

.....

توقيع اللجنة

.....

توقيع رئيس الدائرة

.....

Dedication

To our fathers to our mothers

To our sisters and our brothers

To all our teachers

To Palestine Polytechnic University

*To our supervisor who always
encouraged and supported us
Eng. Ayman Wazwaz*

To all our families and to all our friends

To Palestine the sanctify land

ACKNOWLEDGMENT

We acknowledge Palestine Polytechnic University for giving us the possibility to show some of what we have learned from it.

And we acknowledge all the instructors in the electrical and computer department for their great impact in our education, especially the supervisor of this project eng. Ayman wazwaz ,eng. Sami Salamin, and everybody who effected our education.

Finally we can't forget to acknowledge our great parents who scarified themselves for educating us and facilate our life, and for all their coffee and tolerance.

Manar Ikhlayile
Sabha Abu Sabha
Yasmin Al-Shamesti

Abstract

Mobile phone can serve as powerful tool for world-wide communication. A system will be developed to remotely monitor process through sending messages using mobile.

This system is based on GSM network technology for transmission of messages (Multimedia Messaging Service MMS, Short Message Service SMS) between system and owner.

The system maintains the security alert which is achieved in a way that on the detection of any abnormal events, the system allows automatic generation of messages thus alerting the user against security risk.

Sensors will detect any abnormal events happens inside the home, such that gas leaking, fire, and outer threats, send signal to microcontroller , after that PIC make some processing of received signal, then transmit to it to PC where makes some local analysis in order to turn event's room cameras on . The camera will captures a stream of images and videos ,transmit to PC. PC then store captured images and videos in a file for later owner usage, PC then send an image to peer mobile in addition to useful information about system state.

Peer mobile sends MMS message containing event image, plus a text message containing house state information, then send to Terminal Mobile carried by owner. This Message will tell the owner what happened in his home by using specific control word written in the text part also it's provide the owner an image which express to him event nature occur inside his home.

Table Of Content

Number	Subject	Pages
-	Dedication	v
-	Acknowledgements	vi
-	Abstract	vii
-	Table of Content	viii
-	List of Tables	xi
-	List of Figure	xiv
-	References	xv
-	Appendix	xvi
Chapter one		
1.1	Overview	2
1.2	Problem Statement	2
1.3	Project Objective	3
1.4	Project Benefits	3
1.5	Literature Review	3
1.6	Estimated Cost	4
1.7	Time Planning	5
1.8	Report Content	7
Chapter two		
2.1	Introduction	9
2.2	Sensing Unit	0
2.2.1	Sensors	10
2.2.2	Sensor type	11
2.3	Monitoring System	28
2.4	Mobile Phone	29
2.4.1	GSM Technology	30
2.4.2	SMS	32

2.4.4	Bluetooth	35
2.4.5	Mobile Programming	36

Chapter three

3.1	Introduction	41
3.2	Project Functions	41
3.3	System Components	42
3.4	Sensing Unit	43
3.5	Monitoring unit	45
3.6	Mobile Unit	48
3.7	How system works	52
3.8	System Flow chart	54

Chapter Four

4.1	Introduction	56
4.2	Sensing unit	57
4.3	Monitor unit	59
4.4	Mobile unit	71

Chapter Five

5.1	Introduction	76
5.2	Software Requirements Specification	76
5.2.1	Software Description of sensor Interface	76
5.2.2	Software Description of cameras	83

Chapter Six

6.1	Introduction	92
6.2	Sensing unit	92
6.3	Monitoring unit	96
6.4	Mobile Unit	99

6.5	Sign Midlet	110
Chapter Seven		
7.1	Introduction	113
7.2	System Achievements	113
7.3	Real Learning Outcomes	113
7.4	Recommendation	114

List of Figure

Number	Figure	Pages
1.1	Remote Monitoring Home	2
2.1	Functional elements of measurement	10
2.2	Pins Out of PIC 18F4550	17
2.3	Block Diagram Of The A/D Module	19
2.4	Pins Out Of The RS232 Connector	23
2.5	Working Of RS232 level Converter	24
2.6	Simplified Transmission block diagram	25
2.7	Simplified reception block diagram	25
2.8	GSM Network	31
3.1	System Module Structure	42
3.2	Distributed Home sensors	44
3.3	Sensors to microcontrollers to PC block diagram	45
3.4	TEAC Webcam	46
3.5	Cameras to PC Block diagram	47
3.6	Mobile Block Diagram	49
3.7	Mobile PC interface	50
3.8	Sending An SMS message	51
3.9	Sending An MMS message	51
3.10	System Flow Chart	54
4.1	General System Interfacing Block Diagram	56

4.2	Minimum Requirements for PIC to Run	57
4.3	Crystal Operation	59
4.4	Type Of Sensors	61
4.5	Block Diagram Of LM35	62
4.6	interface LM35 sensor to PIC18F4550	65
4.7	Interface LM35, Motion to PIC 18F4550	66
4.8	Interface Smoke to PIC	67
4.9	All System Interface	69
4.10	State Diagram For Monitoring Unit	71
4.11	State Diagram For Mobile Unit	72
4.12	MIDlet Main Class Methods	73
4.13	Nokia 2690	74
5.1	MPLAB Environment	78
5.2	Memory Usage Gauge	78
5.3	Step A/D Conversions	80
5.4	USART Library	81
5.5	Flow Chart Sensing Unit	82
5.6	USB Webcam	83
5.7	Video format dialog	85
5.8	Video source dialogs	86
5.9	Monitoring Flow Chart	87
5.10	Mobile Flow Chart	91
6.1	Interface Magnet To PIC	93

6.2	Interface To MAX232	94
6.3	Connection Name	95
6.4	Configuration Of Port	95
6.5	Status On Terminal	96
6.6	Interface To Serial Port	96
6.7	Camera Device	97
6.8	Video Source Dialog	98
6.9	Camera Form Program	98
6.10	Mobile Paired Device	99
6.11	SMS On Simulator	100
6.12	Image on Simulator	101
6.13	Mobile Testing Result	102
6.14	Mobile Root Problem	104
6.15	PC root	105
6.16	Mobile Roots	105
6.17	Application On Mobile Screen	106
6.18	Write Authentication Problem	107
6.19	Read Authentication Problem	107
6.20	Send MMS Authentication Problem	108
6.21	Not Trusted Application Problem	108
6.22	Phone Application Access	109
6.23	Application Data Access	109
6.24	Data Access Options	110

6.25	Data Access Option 1	110
6.26	Data Access Option 2	111

List Of Tables

Number	Table	Pages
1.1	Hardware costs	5
1.2	Software Cost	5
1.3	Schedule Table (1'st Semester)	6
1.4	Schedule Table (2'st Semester)	6
2.1	RS232 voltage value	22
2.2	RS232 cable length	22

1

Chapter One

Introduction

- 1.1 Overview**
- 1.2 Problem Statement**
- 1.3 Project Objective**
- 1.4 Project Benefits**
- 1.5 Literature Review**
- 1.6 Estimated Cost**
- 1.7 Time Planning**
- 1.8 Report Content**

2

Chapter Two

Theoretical Background

- 2.1 Introduction**
- 2.2 Sensing Units**
- 2.3 Monitor unit**
- 2.4 Mobile Unit**

3

Chapter Three

Design Concept

- 3.1 Introduction**
- 3.2 Project Functions**
- 3.3 System Components**
- 3.4 How System Work**
- 3.5 System Flow Chart**

4

Chapter Four

Hardware System Design

4.1 Introduction

4.2 Sensing Unit

4.3 Monitor Unit

4.4 Mobile Unit

5

Chapter Five *Software System Design*

5.1 Introduction

5.2 software Requirement specification

6

Chapter Six

Testing and Problems

6.1 Introductions

6.2 Sensing unit

6.3 Monitoring unit

6.4 Mobile unit

6.5 Interface part testing.

7

Chapter Seven

Conclusion and Recommendation

- 7.1. Introduction
- 7.2. System Achievements
- 7.3 Real Learning Outcomes
- 7.4. Recommendation

1.1 Overview

Nowadays, one may stay far away from a home for a long time, and because of this and in order to keep home secure, we need a system using technology tools such as mobile which can be used to achieve home protection 24 hours a day, the ability to remotely monitor household devices using a mobile system has a great advantages for the home-owners. This system provide the owners a sufficient information about any abnormal events happen inside the home.

1.2 Problem Statement

Persons living in an environment affected and influenced by, resulting from their interactions in this world several natural needs and most important one is to know what is happening in the house during their absence .For example, someone may want to keep his home away from fire, gas leaking, or prevent any person tempering with the contents of home.

So how we can keep our homes safe and under monitoring at all time? We want to use mobile technology and to take advantage of mobile services (Multimedia Messaging Service MMS ,Short Message Service SMS , ...) in order to try find a possible solution of these difficulties, figure 1.1 illustrate this process.

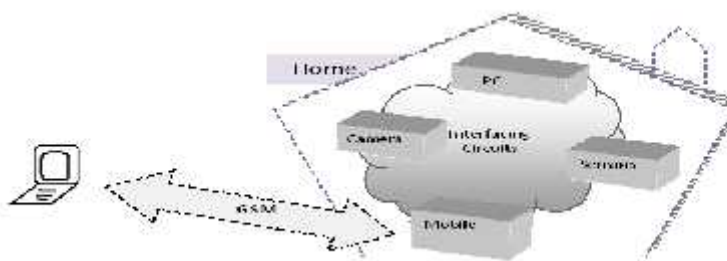


Figure 1.1 Remote Monitoring Home

As we see in Figure 1.1, we have two peer mobile phones communicate directly with each other using GSM technology to contact home continuously, also we have hardware controlling system (microcontroller) which communicates with sensor in home , and a cameras to take picture of events immediately when it occurs and send this picture to a computer connected to these cameras , then send images via (MMS) in addition of proper text message to the owner mobile phone. These components are linked to each other with the appropriate communication channels.

1.3 Project Objective

This project mainly aims to safety and monitor the house 24 hours a day using a mobile, depends on the data sent to the owner of the house when an event occurs.

1.4 Project Benefits

There are several benefits from this project:

1. Maintain the integrity of the house from the outside threats (security system).
2. Utilizing modern technology.
3. This project can be used in developing smart houses.

1.5 Literature review

Many projects have been developed in systems monitoring home, but few had used MMS technology. An example is the project entitled by "Home Security Using Mobile Phone"^[1], which used GSM Control Panel System adopts the newest GSM

network and Digital Signal Processing technology, and is widely used in security field With SMS data transmission.

Another similar project title “Home security”^[2], was developed at Alnajah university, control system allows the owner of the house in which confidence about the state of his house any time he wants through his account on the Web site of that system gives every homeowner an account inside it can access to his page (for his home) and see the house directly by camera system distributor within home.

We provide several differences from previous projects, in case of reach more usable system which alert the user directly at any time (real time).

Our systems discriminate by sending an image to the user in shape of MMS messages represent the nature of the events.

Also differ in form that this system stays far away from traditional audio system which may has defects, then try not to alarm thief about system existence.

1.6 Estimated Cost

The following table shows the component used in project ; which estimated hardware and software costs are presented.

1.6.1 Hardware Cost

The following table shows estimated hardware costs.

Table 1.1 hardware costs.

1	Microcontroller 18F4550	\$10 * 1
2	Smoke Sensor	\$40 * 1
3	Motion Sensor	\$30 * 1
4	Magnet Sensor	\$10 * 1
5	Temperature Sensor	\$5 * 1
6	Camera	\$30 * 2
7	Mobile	\$150 * 1
8	Peripherals	\$ 50
9	Documentation	\$50
6	Total	\$400

1.6.2 Software Cost

The following table shows estimated software costs.

Table 1.2 Software Cost.

1	Java supported ID (Net Beans)	\$10
2	Microchip MPLAB	\$80
	Total	\$90

1.7 Time Schedule

The following table explains the expected timing plan.

First semester

Table 1.3 Schedule Table.

Tasks \ Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
System Definition	█	█	█	█	█	█										
Requirement Analysis							█	█								
System Design									█	█	█	█	█	█		
Documentation									█	█	█	█	█	█	█	█

Second semester

Table 1.4 Schedule Table.

Tasks \ Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
System Design	█	█	█													
Development Phase				█	█	█	█	█	█	█	█					
Test Phase											█	█	█	█	█	
Documentation											█	█	█	█	█	

1.8 Report Content

The report is divided into seven chapters; First chapter is the introduction, which describes the general review of the system, project requirement (Hardware and Software), system cost, scheduling time, and report contents.

Chapter two discusses the theoretical background that shows the theoretical subjects related to the main ideas of the project information about mobile unit, control unit, sensing unit and other special component that are built to distinguish the system.

Chapter three talk about design concepts; the main objectives and a general block diagram that shows how the system works.

Chapter four discusses hardware design that describes system. Chapter five discusses software design that describes system, chapter six discusses testing for system and some problem and challenges meet in our project.

Finally chapter seven discusses Conclusion and recommendation.

2.1 Introduction

Embedded systems have grown tremendously in recent years not only in their popularity but also in their complexity. Gadgets are increasingly becoming intelligent and autonomous. Refrigerators, air-conditioners, automobiles, mobile phones etc are some of the common examples of devices with built-in intelligence. These devices function based on operating and environmental parameters.

This project is divided mainly into three parts:

- **Sensing Unit:**
To detect any risk using sensors, and notified other component to start working.
- **Monitor Unit :**
To transmit nature of an event to PC through microcontroller on one hand, and inform camera to capture images and video on other hand.
- **Mobile Unit :**
How to connect mobile with PC, and how to take an image from PC to send it in form MMS image through mobile using GSM technology.

2.2 Sensing Unit

Sensors are used to gather information to provide data to give an understanding of the current status of the system parameters. In a control system, the signal from the sensor is input to a controller . The controller then provides an output to govern the measured parameters as shown in the figure 2.1.

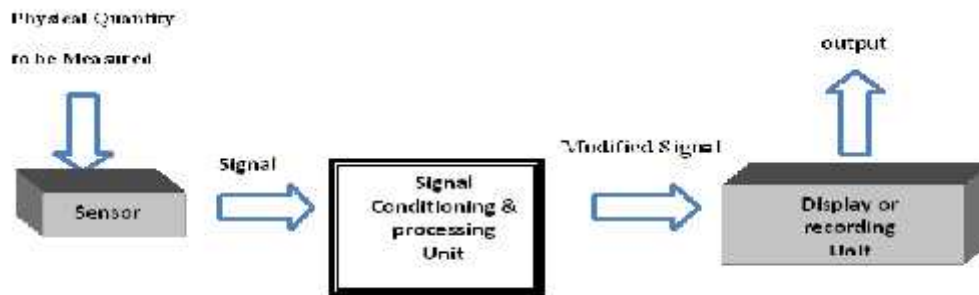


Figure 2.1 Functional elements of measurement

In this System there's a need for sensors to sense different variables or thing and then return information or data to the user. This part explains what is sensor and some type of sensor and how they work to complete their functions.

2.2.1 Sensors:

A sensor is a device that converts a physical phenomenon into an electrical signal. In recent years, enormous capability for information processing has been developed within the electronics industry. The most significant example of this capability is the personal computer. In addition, the availability of inexpensive microcontroller is having a tremendous impact on the design of embedded computing products ranging from automobiles to microwave ovens to toys. In recent years, versions of these products that use microcontroller for control of functionality are becoming widely available. In automobiles, such capability is necessary to achieve compliance with pollution restrictions. In other cases, such capability simply offers an inexpensive performance advantage^[3].

All of these microcontrollers need electrical input voltages in order to receive instructions and information. So, along with the availability of inexpensive microcontroller has grown an opportunity for the use of sensors in a wide variety of products. In addition, since the output of the sensor is an electrical signal, sensors

tend to be characterized in the same way as electronic devices. The data sheets for many sensors are formatted just like electronic product data sheets.

However, there are many formats in existence, and there is nothing close to an international standard for sensor specifications. The system designer will encounter a variety of interpretations of sensor performance parameters, and it can be confusing. It is important to realize that this confusion is not due to an inability to explain the meaning of the terms—rather it is a result of the fact that different parts of the sensor community have grown comfortable using these terms differently.

2.2.2 Sensor Type:

1- Temperature sensor



Temperature sensors detect a change in a physical parameter such as resistance or output voltage that corresponds to a temperature change.

- **LM35 Precision Centigrade Temperature Sensors :**

In this project use LM35 to indication about fire occur in the home.

General Description:

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The system needs this sensor to achieve one of its main targets to sense if fire occurs by measure the temperature.

Features

- a) Calibrated directly in ° Celsius (Centigrade)
- b) Linear + 10.0 mV/°C scale factor
- c) 0.5°C accuracy guarantee able (at +25°C)
- d) Rated for full –55° to +150°C range
- e) Suitable for remote applications
- f) Operates from 4 to 30 volts
- g) Less than 60 µA current drain
- h) Low self-heating, 0.08°C in still air
- i) Nonlinearity only ±1/4°C typical
- j) Low impedance output, 0.1 W for 1 mA load

2- Motion detector (PARADOX PRO 476+)



A motion detector is a device that contains a physical mechanism or electronic sensor that quantifies motion that can be either integrated with or connected to other devices that alert the user of the presence of a moving object within the field of view. They form a vital component of comprehensive security systems, for both homes and businesses.

Motion detector contains a motion sensor that transforms the detection of motion into an electric signal. This can be achieved by measuring optical or acoustical changes in the field of view. Most motion detectors can detect up to 50-80 feet. A motion detector may be connected to a burglar alarm that is used to alert the home owner or security service after it detects motion. Such a detector may also trigger a red light camera.

3- Smoke detector :



A smoke detector is a device that detects smoke, typically as an indicator of fire. Commercial, industrial, and mass residential devices issue a signal to a fire alarm system, while household detectors, known as smoke alarms, generally issue a local audible and/or visual alarm from the detector itself.

Smoke detectors are typically housed in a disk-shaped plastic enclosure about 150 millimeters (6 in) in diameter and 25 millimeters (1 in) thick, but the shape can vary by manufacturer or product line. Most smoke detectors work either by optical detection (photoelectric) or by physical process (ionization), while others use both detection methods to increase sensitivity to smoke. Sensitive alarms can be used to detect, and thus deter, smoking in areas where it is banned such as toilets and schools. Smoke detectors in large commercial, industrial, and residential buildings are usually powered by a central fire alarm system, which is powered by the building power with a battery backup. However, in many single family detached and smaller multiple family housings, a smoke alarm is often powered only by a single disposable battery.

4- Door And Windows Sensors

Some home security system door sensors are wired to an electric circuit within the house, and other home security systems today are wireless. However, whether they are wired or wireless, a home security system consists of motion detectors and sensors that are placed on the doorway and often the windows. Magnetic sensors are installed to secure the most susceptible exterior doorway and window openings. When an intruder enters your home by opening a door or window , the monitoring station is instantly alerted.

Magnets Switches :



The magnet on the sensor keeps voltage flowing through the circuit area whenever the doors and windows are closed. However, when the system is activated and a door or window is opened, there is a break in the flow of the voltage to the circuit, and this causes the activation of the alarm. The security monitoring agency and police are quickly notified once this occurs. Modern-day security systems are actually capable of alerting the monitoring station of which door or window in particular has been breached.

2.2.2 Microcontroller

A microcontroller is “a small computer on a single integrated circuit consisting of a relatively simple CPU combined with support functions such as a crystal oscillator, timers, serial and analog I/O etc. The name PIC was originally an acronym for "Programmable Intelligent Computer".

Microcontrollers have long been a convenient interface for embedded systems; they represent the core of the control system for electronic devices in dedicated applications. Thus, in contrast the microprocessors that are used in general purpose applications like personal computers that need high-performance, and multitasking. Microcontrollers contain data and program memory, serial and parallel I/O, timers, external and internal interrupts, and peripherals. These make them a strong choice when implementing control systems.

The PIC chips (or PICmicro chips) are as stated above, programmable chips, programmed to perform a special operations for dedicated applications in embedded

systems, they provide strong interfacing abilities with many peripherals and other microcontrollers in other embedded systems.

Programming PIC microcontrollers is a simple three steps process, write the code, compile the code, and upload the code into a microcontroller. Writing the code can be developed in many Integrated Development Environments (IDE's) for example, MPLAP IDE, which is software developed for the Microchip appliances like the PIC microcontrollers. Compiling the code can be done by the compiler of the MPLAP IDE. There are different compilers associated to work with PIC chips, C compiler, or assembler for assembly language codes, and many more. The decision of which compiler to use, is a developer choice, depending on the application which the PIC is a part of. The final step of programming the PIC chip is uploading the code into the microcontroller.

The PIC microcontroller architecture makes interfacing most peripherals with the PIC a far from hard task, the I/O's are organized as ports, PORTA, PORTB, etc. each port can be treated as a unit or as single I/O pins, 8-bit, 6-bit or other organization. Each port was initially configured to do a specific operation, serial data operations, Analog-to-Digital conversion, and many more, but it's not always necessary to stick with the initial configuration, each port or single I/O pin can be configured to do different operation from what initially configured

In this project the PIC18F4550 will be used. This is due to its availability, cheap cost. The PIC18F4550 has all what is needed for the implementation of this project, enough I/O Ports, ADC, timers and counters, and operating frequency of 20MHz^[4]

PIC18F4550 peripherals :



The PIC18F4550 have 5 port A,B,C,D, and E and it described as follows :

- Each port have three register associated for the his operation and these register is PORTX ,LATX, and TRISX.
- The port data itself appear in the PORTX.
- The data direction (input or output) is determined by the bit values set in the (control register) TRISX.
- The LATX is used to read the output data back (not input read) .

1- PORTA

- The port can be used for general purpose bi directional digital data. It is also shared with the analog function .
- The Timer 0 input is shared with bit 4 of the port.
- The three register associated with port A – PORTA,TRISA, and LATA.

2- PORTB

- Three primary register PORTB,TRISB, and LATB.
- That bits 5-7 share with the in-circuit debug function of PGC ,PGD and PGM.
- The external interrupt can be made through PORTB pins .

3- PORTC

- Its Three primary register PORTC,TRISC, and LATC.
- This port share with serial,... .

4- PORTD

- Parallel slave itself , with 3 bits of port E for handshake .

Some peripheral such as USART , analog to digital converter , watch dog timer , and low voltage detect will explain later ,and other peripheral not need such as timers (our PIC have 5 programmable timers), capture/compare/pulse width modules. for other information for any part can see the data sheet in appendix .

In figure 2.2 we show the pins of PIC 18F4550 , and in chapter 4 we learn more about these pins and used it in our project.

40-Pin PDIP

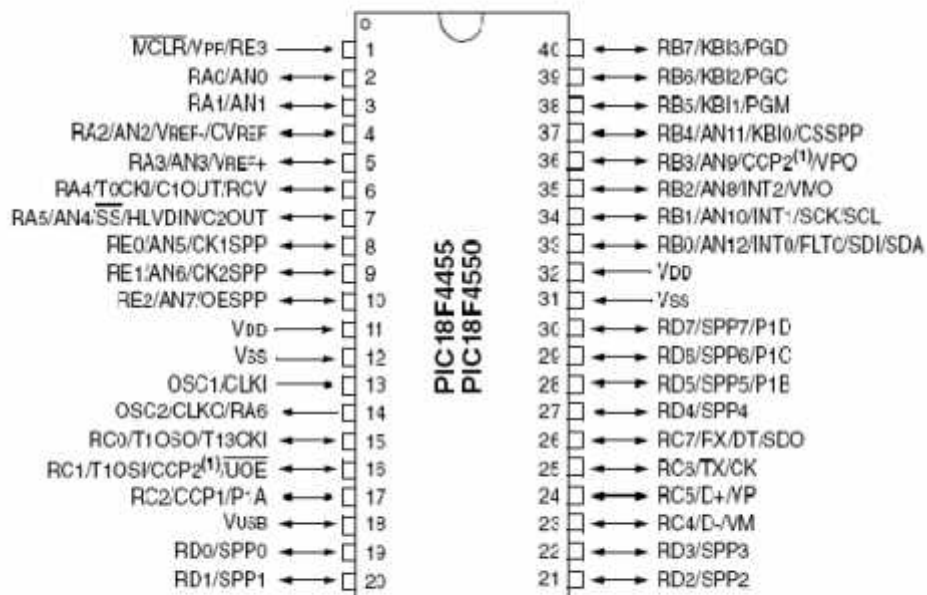


Figure2.2 Pins out of PIC18F4550.

Analog-to-Digital converter :

This converter built inside PIC . the PIC has 10-Bit ADC (A/D) Module , 13 pin (PIC 18F4550) for analog input with multiplexer . This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (VDD and VSS) or the voltage level on the A3/AN3/VREF+ and RA2/AN2/VREF-/CVREF pins.

Each port pin associated with the A/D converter can be configured as an analog input or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/DONE bit (ADCON0 register) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 2.3 .

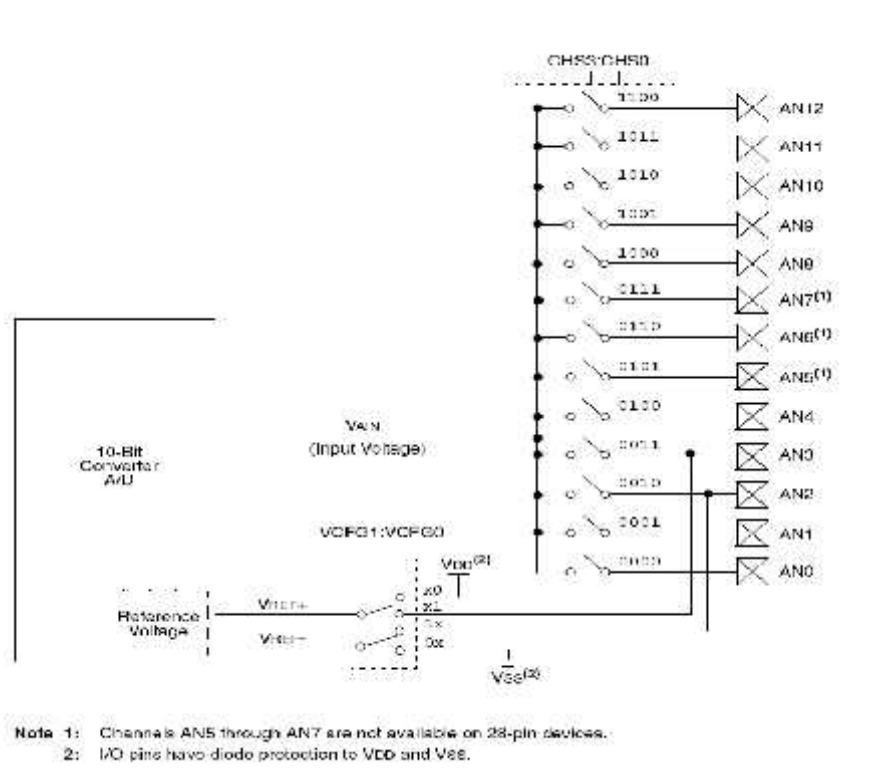


Figure 2.3 Block diagram of the A/D module

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/DONE bit and the actual start of the conversion. In chapter four we show how embedded this module on our work.

Acquisition Time

When an specific channel (AN0 for example) is selected the voltage from that input channel is stored in an internal holding capacitor. It takes some time for the capacitor to get fully charged and become equal to the applied voltage. This time is called acquisition time. The PIC18F4550's ADC provides a programmable

acquisition time, so you can setup the acquisition time. Once acquisition time is over the input channel is disconnected from the source and the conversion begin. The acquisition times depends on several factor like the source impedance, Vdd of the system and temperature. Can refer to the datasheet for details on its calculation. A safe value is 2.45uS, so acquisition time must be set to any value more than this.

ADC Clock

ADC Requires a clock source to do its conversion, this is called ADC Clock. The time period of the ADC Clock is called T_{AD} . It is also the time required to generate 1 bit of conversion. The ADC requires 11 T_{AD} to do a 10 bit conversion. It can be derived from the CPU clock (called T_{OSC}) by dividing it by a suitable division factor. There are Seven possible option.

2 x T_{OSC}

4 x T_{OSC}

8 x T_{OSC}

16 x T_{OSC}

32 x T_{OSC}

64 x T_{OSC}

Internal RC

For Correct A/D Conversion, the A/D conversion clock (T_{AD}) must be as short as possible but greater than the minimum T_{AD} .refer to datasheet it is 0.7uS for PIC18FXXXX device.

Serial Port :

In order to make two devices communicate, whether they are desktop computers, microcontrollers, or any other form of integrated circuit, we need a method of

communication and an agreed-upon language. The most common form of communication between electronic devices is *serial communication*. Communicating serially involves sending a series of digital pulses back and forth between devices at a mutually agreed-upon rate. The sender sends pulses representing the data to be sent at the agreed-upon *data rate*, and the receiver listens for pulses at that same rate.

Communication as defined in the **RS232** standard is an asynchronous serial communication method. The word *serial* means, that the information is sent one bit at a time. *Asynchronous* tells us that the information is not sent in predefined time slots. Data transfer can start at any given time and it is the task of the receiver to detect when a message starts and ends

RS232 physical properties

The RS232 standard describes a communication method capable of communicating in different environments. This has had its impact on the maximum allowable voltages etc. on the pins. In the original definition, the technical possibilities of that time were taken into account. The maximum baud rate defined for example is 20 kbps. With current devices like the 16550A UART, maximum speeds of 1.5 Mbps are allowed.

Voltages

The signal level of the RS232 pins can have two states. A high bit, or mark state is identified by a *negative* voltage and a low bit or space state uses a *positive* value. This might be a bit confusing, because in normal circumstances, high logical values are defined by high voltages also. Table 2.1 show the voltage limits.

Table 2.1 RS232 voltage value

RS232 voltage values		
Level	Transmitter capable (V)	Receiver capable (V)
Space state (0)	+5 ... +15	+3 ... +25
Mark state (1)	-5 ... -15	-3 ... -25
Undefined	-	-3 ... +3

Maximum cable lengths

Cable length is one of the most discussed items in RS232 world. The standard has a clear answer, the maximum cable length is 50 feet, or the cable length equal to a capacitance of 2500 pF. The latter rule is often forgotten. This means that using a cable with low capacitance allows you to span longer distances without going beyond the limitations of the standard. If for example UTP CAT-5 cable is used with a typical capacitance of 17 pF/ft, the maximum allowed cable length is 147 feet. In the following table 2.2 show RS232 cable length^[5] .

Table 2.2 RS232 cable length

Baud rate	Maximum cable length (ft)
19200	50
9600	500
4800	1000
2400	3000

In the following, figure 2.4 show the pin out of the RS232 connector.

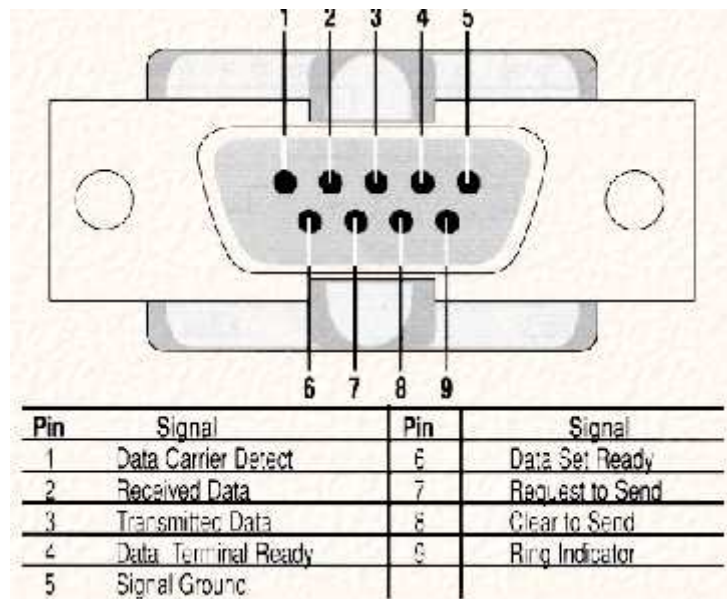


Figure 2.4 pin out of the RS232 connector

MAX232 Level Converter:



In the last pages we saw that how RS232 level signals differs from normal logic signals. So to interface RS232 level signals to our MCUs we need a "Level converter". Can use Max232 .

What a level converter will do ? as show in figure 2.5 max232 convert RS232 level signals (HIGH=-12V LOW=+12V) from PC to TTL level signal (HIGH=+5V LOW=0V) to be fed to MCU and also the opposite.

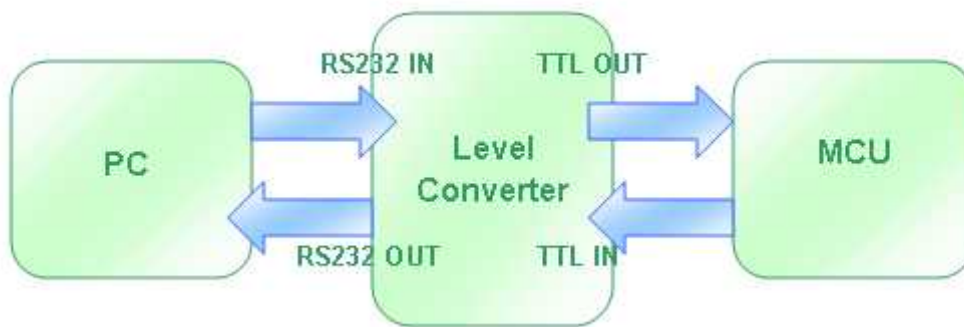


Fig 2.5 Working of RS232 level converter

The USART:

USART stands for Universal Synchronous Asynchronous Receiver Transmitter. It is sometimes called the Serial Communications Interface or SCI.

Synchronous operation uses a clock and data line while there is no separate clock accompanying the data for Asynchronous transmission. Since there is no clock signal in asynchronous operation, one pin can be used for transmission and another pin can be used for reception. Both transmission and reception can occur at the same time this is known as full duplex operation.

Transmission and reception can be independently enabled. However, when the serial port is enabled, the USART will control both pins and one cannot be used for general purpose I/O when the other is being used for transmission or reception. The USART is most commonly used in the asynchronous mode. In this presentation we will deal exclusively with asynchronous operation. The most common use of the USART in asynchronous mode is to communicate to a PC serial port using the RS-232 protocol. The USART can both transmit and receive.

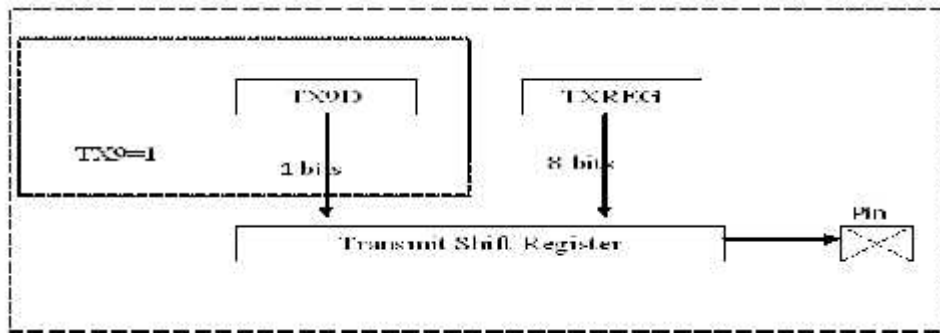


Figure 2.6 Simplified transmission block diagram

As shown in figure 2.6 , the USART can be configured to transmit eight or nine data bits by the TX9 bit in the TXSTA register. If nine bits are to be transmitted, the ninth data bit must be placed in the TX9D bit of the TXSTA register before writing the other eight bits to the TXREG register. Once data has been written to TXREG, the eight or nine bits are moved into the transmit shift register. From there they are clocked out onto the TX pin preceded by a start bit and followed by a stop bit.

The use of a separate transmit shift register allows new data to be written to the TXREG register while the previous data is still being transmitted. This allows the maximum throughput to be achieved.

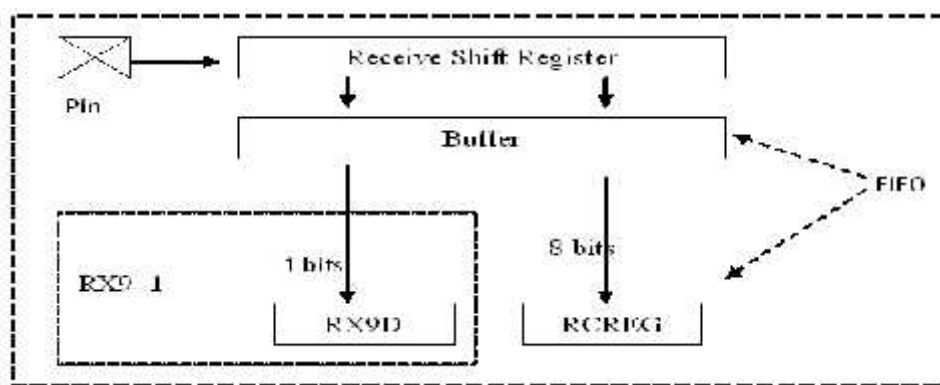


Figure 2.7 Simplified reception block diagram

As shown in figure 2.7, the USART can be configured to receive eight or nine bits by the RX9 bit in the RCSTA register. After the detection of a start bit, eight or nine bits of serial data are shifted from the RX pin into the receive shift register one bit at a time. After the last bit has been shifted in, the stop bit is checked and the data is moved into the buffer which passes the data through to the RCREG register if it is empty. The buffer and RCREG register therefore form a two element FIFO. If nine bit reception is enabled, the ninth bit is passed into the RX9D bit in the RCSTA register in the same way as the other eight bits of data are passed into the RCREG register.

The use of a separate receive shift register and a FIFO buffer allows time for the software running on the PICmicro MCU to read out the received data before an overrun error occurs. It is possible to have received two bytes and be busy receiving a third byte before the data in the RCREG register is read.

The USART outputs and inputs logic level signals on the TX and RX pins of the PICmicro MCU. The signal is high when no transmission or reception is in progress and goes low when the transmission starts. This low going transition is used by the receiver to synchronize to the incoming data. The signal stays low for the duration of the start bit and is followed by the data bits, least significant bit first. In the case of an eight-bit transfer, there are eight data bits and the last data bit is followed by the stop bit which is high. The transmission therefore ends with the pin high. After the stop bit has completed, the start bit of the next transmission can occur as shown by the dotted lines. There are several things to note about this waveform, which represents the signal on the TX or RX pins of the microcontroller. The start bit is a zero and the stop bit is a one. The data is sent least significant bit first so the bit pattern looks backwards in comparison to the way it appears when written as a binary number. The data is not inverted even though RS-232 uses negative voltages to represent a logic one. Generally, when using the USART for RS-232

communications, the signals must be inverted and level shifted through a transceiver chip of some sort.

The signals on the USART pins of the microcontroller use logic levels. This means that for a five volt supply, the signals will be close to five volts when they are high and close to ground when they are low. When communicating with other logic devices, these signals can be used directly. In many applications, particularly with asynchronous communications, transmission standards such as RS-232 and RS-485 require different voltage levels to be used. For example, RS-232 uses a voltage below minus five volts to represent a logic one and a voltage above five volts to represent a logic zero. For RS-232, an interface chip such as Microchip's max232 device is recommended to convert the signals to the required levels.

There are several registers used to control the USART .The SPBRG register allows the baud rate to be set. The TXSTA and RCSTA registers are used to control transmission and reception but there are some overlapping functions and both registers are always used. The TXREG and RCREG registers are used write data to be transmitted and to read the received data.

The rate at which data is transmitted or received must be always be set using the baud rate generator unless the USART is being used in synchronous slave mode. The baud rate is set by writing to the SPBRG register. The SYNC bit selects between synchronous and asynchronous modes, and these modes have different baud rates for a particular value in the SPBRG register. For asynchronous mode, the SYNC bit must be cleared and the BRGH bit is used to select between high and low speed options for greater flexibility in setting the baud rate.

Formulas for baud rate

Baud rate = $F_{osc}/(16(SPBRG+1))$, BRGH=1 2.1

Baud rate = $F_{osc}/(64(SPBRG+1))$, BRGH=0 2.2

Formulas for SPBRG

$$\text{SPBRG} = (\text{Fosc}/(16 \times \text{Baud rate})) - 1, \text{BRGH}=1 \dots\dots\dots 2.3$$

$$\text{SPBRG} = (\text{Fosc}/(64 \times \text{Baud rate})) - 1, \text{BRGH}=0 \dots\dots\dots 2.4$$

The top two formulas show how the baud rate is set by the value in the SPBRG register and the BRGH bit. More important for the user, however, is to be able to calculate the value to place in the SPBRG register to achieve a desired baud rate. The bottom two formulas can be used to do this. The SPBRG register can have a value of zero to 255 and must always be an integer value. When these formulas yield a value for SPBRG that is not an integer, there will be a difference between the desired baud rate and the rate that can actually be achieved. By calculating the actual baud rate using the nearest integer value of SPBRG, the error can be determined. Whether this error is acceptable usually depends on the application.

2.3 Monitoring System

2.3.1 Camera

Camera is the primary monitoring element, it is responsible for viewing and send stream of video to the computer. To choose a specific camera we should take in consideration some features including zoom and mega pixels.

Another important element for choosing camera is the connection type, where the camera can be connected via USB or other computer port. Choosing the cameras depends on the system critical needs and the available resources to buy this camera.

In our system we choose a PC web cam to get continues stream which will framed in the system and deal with to extract needing information.

2.3.2 Light Source

To get enough light for monitoring system you can provide two ways:

- Using Special type of cameras which takes night-capturing.
- Using additional electrical light source than the regular room-light.

You can get the best performance by using both techniques. But it is expensive to buy a night-capturing camera. So we apply the second one only. Physically, we will need different light sources to be placed in the corners or in the room.

2.3.4 Cameras locations

Must take into account the distribution of places cameras to capture an appropriate image which contain all features required.

2.3.5 Camera and Connection

In the narrow-scale monitoring system we use two or three camera to capture images, send it using wires to USB.

2.4 Mobile Phone

Mobile phone is one of the project basic elements, it will be used to communicate with PC and with another such mobile, in order to receive data. Cell phone also called hand phone is an electronic device used for mobile telecommunications over a cellular network of specialized base stations known as cell sites. Most current cell phones connect to a cellular network consisting of switching points and base stations

(cell sites) owned by a mobile network operator. A cell phone is a **full-duplex** device.

Channels: A typical cell phone can communicate on 1,664 channels or more.

Range: The phone have a low –power transceiver that transmits voice and data to the nearest cell sites ,usually 5 to 8 miles(about 8 to 13 kilometers)away; Because cell phones operate within **cells**, when the cellular phones is turned on, it registers with the mobile telephone exchange ,or switch ,with its unique identifiers , and they can switch cells as they move around. Cells give cell phones incredible range. Someone using a cell phone can drive hundreds of miles and maintain a conversation the entire time because of the cellular approach^[6] .

Our mobile will be choose to fit the project objectives such as serial/ parallel communication, wireless/ Bluetooth technologies and must support the AT commands for the short message service (SMS) and multimedia messaging service. such mobile will be used in our project to communicate either with microcontroller or pc and also to communicate with another such mobile.

2.4.1 GSM Technology

(Global System for Mobile Communications) is the most popular standard for mobile telephony systems in the world. GSM differs from its predecessor technologies in that both signaling and speech channels are digital, and thus GSM is considered a second generation (2G) mobile phone system. The existence of GSM everywhere, has been advantage to customer because they can travelling and roaming without needed to change mobile as shown in figure 2.8.

GSM also led in low-cost implementation of the short message service (SMS), GSM networks operate in a number of different carrier frequency ranges (separated into GSM frequency ranges for 2G and UMTS frequency bands for 3G), most 2G GSM networks operating in the 900 MHz or 1800 MHz bands, in some country such as United state and Canada the 850 MHz and 1900 MHz bands were used instead and most 3G **GSM** networks in Europe operate in the 2100 MHz frequency band. In rare cases the 400 and 450 MHz frequency bands are assigned in some countries because they were previously used for first-generation systems.

Regardless of the frequency selected by an operator, it is divided into timeslots for individual phones to use. This allows eight full-rate or sixteen half-rate speech channels per radio frequency which are grouped into a TDMA frame. The channel data rate for all 8 channels is 270.833 Kbit/s, and the frame duration is 4.615 ms. The transmission power in the handset is limited to a maximum of 2 watts in GSM850/900 and 1 watt in GSM1800/1900^[7].

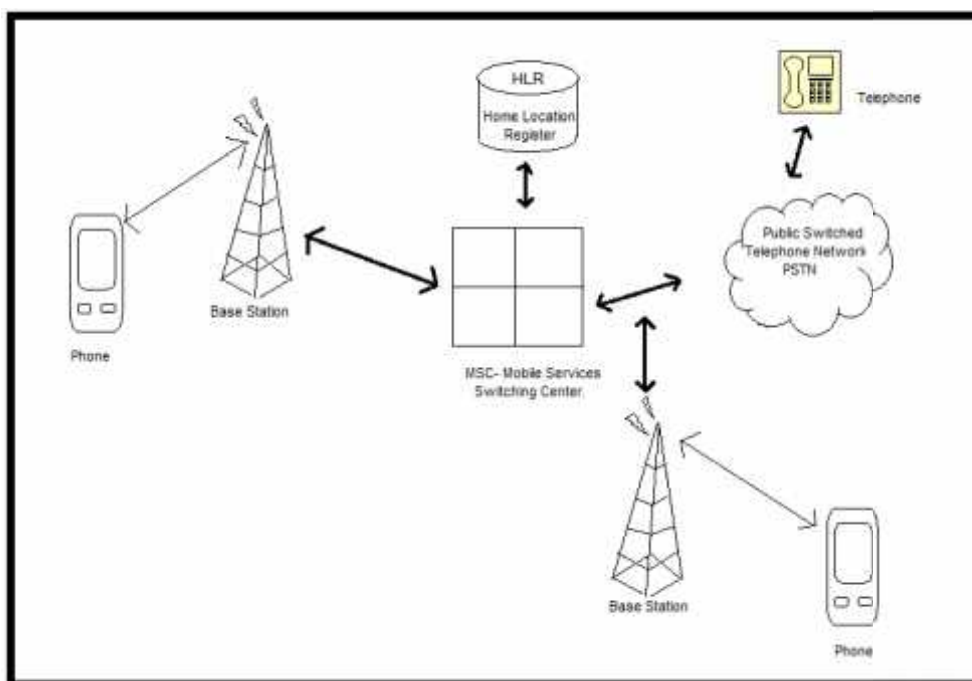


Figure 2.8 GSM Network

2.4.2 SMS

SMS is a method by which messages can be sent to a cell phone via another cell phone, a computer connected to the Internet.

2.4.2.1 SMS In Details

As a communication service component of the GSM mobile communication system; SMS messages may be sent either from one point to another point, or may be sent to all devices within a specific geographical region. The former, known as SMS-PP, is used primarily between individuals communicating within two devices while the latter, known as SMS-CB, may be used to broadcast public announcements. Messages are sent to a Short Message Service Center (SMSC) which provides a "store and forward" mechanism. . It attempts to send messages to the SMSC's recipients. If a recipient is not reachable, the SMSC queues the message for later retry.

Some SMSCs also provide a "forward and forget" option where transmission is tried only once. Both Mobile Terminated (MT, for messages sent *to* a mobile handset) and Mobile Originating (MO, for those sent *from* the mobile handset) operations are supported. Message delivery is "best effort", so there are no guarantees that a message will actually be delivered to its recipient, and delay but complete loss of a message is uncommon.^[8]

2.4.2.2 Message Size

Messages are sent with the MAP (Mobile Application Part) whose payload length is limited by the constraints of the signaling protocol to precisely 140 byte "octets"

(140 octets = 140 * 8 bits = 1120 bits). In practice this translated to either sizes of 160 7-bit characters, 140 8-bit characters, or 70 16-bit characters (including spaces). Characters in languages such as Arabic, Chinese, Korean, Japanese or Cyrillic alphabet languages must be encoded using the 16-bit character encoding. Routing data and other metadata is additional to the payload size. Larger content (Concatenated SMS, multipart or segmented SMS, or "Long SMS") can be sent using multiple messages, in which case each message will start with a user data header (UDH) containing segmentation information.

Since UDH is part of the payload, the number of available characters per segment is lower: 153 for 7-bit encoding, 133 for 8-bit encoding and 67 for 16-bit encoding. The receiving handset is then responsible for reassembling the message and presenting it to the user as one long message. While the standard theoretically permits up to 255 segments, 6 to 8 segment messages are the practical maximum, and long messages are often billed as equivalent to multiple SMS messages ^[9].

2.4.3 MMS

MMS is a standard way to send messages that include multimedia content to and from mobile phones. It extends the core SMS (Short Message Service) capability.

2.4.3.1 MMS In Details

MMS messages are delivered in a completely different way than SMS. The first step is for the sending device to encode the multimedia content. The message is then forwarded to the carrier's MMS store and forward server, known as the MMSC. If the receiver is on another carrier, the relay forwards the message to the recipient's carrier using the Internet. Once the MMSC has received a message, it first

determines if the receiver's supports the standards for receiving MMS. If so, the content is extracted and sent to a temporary storage server with an HTTP front-end.

An SMS "control message" containing the URL of the content is then sent to the recipient's handset to trigger the receiver's WAP browser to open and receive the content from the embedded URL. Several other messages are exchanged to indicate status of the delivery.

Before delivering content, some MMSCs also include a conversion service that will attempt to modify the multimedia content into a format suitable for the receiver. This is known as "content adaptation". If the receiver's handset is not MMS capable, the message is usually delivered to a web based service from where the content can be viewed from a normal internet browser. The URL for the content is usually sent to the receiver's phone in a normal text message.

This behavior is usually known as the 'legacy experience' since content can still be received by a phone number, even if the phone itself does not support MMS. A database is usually maintained by the operator, and in it each mobile phone number is marked as being associated with a legacy handset or not. It can be a bit hit and miss since customers can change their handset at will and this database is not usually updated dynamically^[10].

2.4.3.2 Message Size

Although the standard does not specify a maximum size for a message, 300 kB is the current recommended size used by networks due to some limitations on the WAP gateway side. Some interesting challenges with MMS that do not exist with SMS: Content adaptation: Multimedia content created by one brand of MMS phone may not be entirely compatible with the capabilities of the recipients' MMS phone.

In the MMS architecture, the recipient MMSC is responsible for providing for content adaptation (e.g., image resizing, audio codec ,etc.), if this feature is enabled by the mobile network operator.

Distribution lists: Current MMS specifications do not include distribution lists nor methods by which large numbers of recipients can be conveniently addressed.

Bulk messaging: The flow of peer-to-peer MMS messaging involves several over-the-air transactions that become inefficient when MMS is used to send messages to large numbers of subscribers .

Handset Configuration: Unlike SMS, MMS requires a number of handset parameters to be set. Poor handset configuration is often blamed as the first point of failure for many users^[11] .

2.4.4 Bluetooth

Bluetooth is a wireless personal area network technology (WPAN for short), a low-range wireless network technology used for linking devices to one another without a hard-wired connection Bluetooth devices do not need a direct line of sight to communicate, which makes them more flexible in use and allows room-to-room communication in small spaces.

The aim of Bluetooth is to transmit voice or data between devices with low-cost radio circuits, over a range of about ten to just under a hundred meters, using very little power.

Bluetooth technology is designed mainly for linking devices (such as printers, mobile phones, home appliances, wireless headsets, mice, keyboards, etc.), computers, or PDAs to one another, without using a wired connection. Bluetooth is

also becoming more and more commonly used in mobile phones, allowing them to communicate with computers or PDAs, and is especially widespread in hands-free accessories like Bluetooth headsets. Bluetooth headsets act as advanced earpieces which include remote control features.

Bluetooth is one of today's most exciting technologies. It is a short-range radio wave wireless technology operating in the 2.4 GHz frequency spectrum. With an operating range of 30 feet (10 meters) and a maximum transmission rate of 1Mbps, Bluetooth is widely touted as the "cable replacement" solution^[11]

2.4.5 Mobile Programming

2.4.5.1 Symbian OS

Symbian OS is an operating system designed for mobile devices and smart phones, with associated libraries, user interface and reference implementations of common tools .

2.4.5.1.1 Symbian Design

Symbian features pre-emptive multitasking and memory protection, like other operating systems. Symbian OS was created with three systems design principles in mind:

1. the integrity and security of user data is valuable,
2. user time must not be wasted, and
3. all resources are scarce.

To best follow these principles, Symbian uses a microkernel as we will see later.

The OS is optimized for low-power battery-based devices. The Symbian kernel (EKA2) supports sufficiently-fast real-time response to build a single-core phone around it—that is, a phone in which a single processor core executes both the user applications ; This has allowed Symbian EKA2 phones to become smaller, cheaper and more power efficient than their predecessors .

2.4.5.1.2 Symbian structure

Symbian and as any operating system contains of many layers some of:

- UI Framework layer
- Application Services Layer (Java ME)
- OS Services Layer
- Base Services Layer
- Kernel Services & Hardware Interface Layer.

The Base Services Layer is the lowest level reachable by user-side operations ; in addition of many features contained inside such as File Server and User Library. Symbian has a microkernel architecture(EKA1, EKA2), which means that the minimum necessary is within the kernel to maximize availability and responsiveness. The EKA2 real-time kernel, which has been termed a nanokernel, contains only the most basic primitives and requires an extended kernel to implement any other abstractions. User interface(UI) code is included in symbian with a large volume, the actual user interfaces which were maintained by third parties.

2.4.5.1.3 Some of devices that use Symbian OS

There are many devices which used symbian OS such as: The Ericsson R380 , Sony Ericsson P800, P900, W950 , Nokia 7650. The Nokia N-Gage and Nokia,

Siemens SX1 and Samsung SGH-Z600, the N series (except Nokia N8xx and N9xx), Nokia 9210, 9300 and 9500 .

2.4.5.1.4 Developing on Symbian OS

The native language of Symbian is C++, although it is not a standard implementation. However, Symbian devices can also be programmed using Python, Java ME, Flash Lite, Ruby, .NET, Web Runtime (WRT) Widgets and Standard C/C++. There were multiple platforms based upon Symbian OS that provided SDKs(which are often contained in individual phone) for application developers wishing to target Symbian OS devices. SDKS have the header files and libraries files which needed to get Symbian OS based software.

Once ,Symbian application development finish, everything is needed is to find out an mobile phone to install our application after being packaged in SIS files. As we mentioned, Java ME can be used, Java ME application are developed using some tools such as SUN Java Wireless Toolkit. Super waba is another tools which can also be use to create Symbian programs using Java.

2.4.5.2 Java ME

Java Platform, Micro Edition, j2me, or Java ME, is a Java platform designed for mobile devices and embedded systems with limited memory, display and power capacity. Target devices range from industrial controls to mobile phones .

Java ME was designed by Sun Microsystems; the platform replaced a similar technology, Personal Java. Originally developed under the Java Community Process . Sun provides a reference implementation of the specification, but has tended not to

provide free binary implementations of its Java ME runtime environment for mobile devices, rather relying on third parties to provide their own. Java ME devices implement a *profile*. The most common of these are the Mobile Information Device Profile aimed at mobile devices, such as cell phones, and the Personal Profile aimed at consumer products and embedded devices . Profiles are subsets of *configurations*, of which there are currently two: the Connected Limited Device Configuration (CLDC) using to fit small mobile devices and the Connected Device Configuration (CDC) using with more capable mobile devices like smart-phones, both CLDC and CDC application can be created with Net Beans. Designed for mobile phones, the Mobile Information Device Profile includes a GUI, and a data storage API, and MIDP 2.0 includes a basic 2D gaming API. Applications written for this profile are called MIDlets. Almost all new cell phones come with a MIDP implementation, and it is now the de facto standard for downloadable cell phone games. However, many cell phones can run only those MIDlets that have been approved by the carrier, especially in North America.

3.1 Introduction

Nowadays, the Mobile phones became one of the most important devices which are used by everyone, so the designers must take this point when designing remote control systems. We will try to build a system to achieve the efficiency and usefulness to be a step in the progress of human lives using available technologies.

Our system will use mobile phones to receive data from sensors and cameras. This mobile which directly attached the system will send MMS to another mobile (Terminal mobile) carried by the owner.

3.2 Project Functions

This project aims at achieving the following objectives:

1. Home monitoring 24 hours a day.
2. Home safety and keeping it secured.
3. We will use sensors and cameras to monitor what happens in our house while we are out of home and take actions.
4. Get reports from mobile as remote control and monitor device.
5. There are several ways to use mobile as remote control, like GPRS, SMS, Bluetooth. In this project we will use MMS technologies in order to connect the two mobile phones to each other and take necessary information.
6. Take several images of the home room where the event happen and save them locally on PC then send to mobile via MMS message.
7. Provide high level of security in such part of home where valuable objects exist.

3.3 System Components

In this system the owner must be able to control his home through specified control hardware using two peer mobile phones connecting to each other by GSM network.

As we see in Figure 3.1, the system used a group of components which are connected to each other with an appropriate communication channels, these components can be categorized as following:

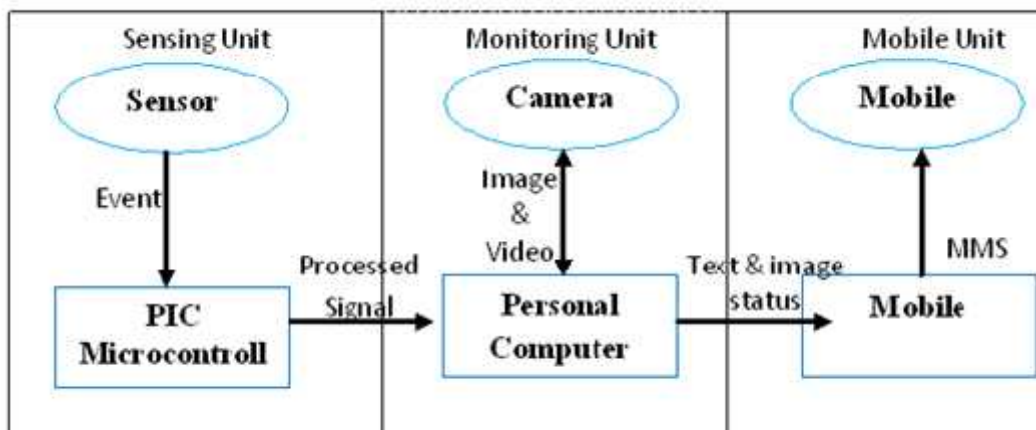


Figure 3.1 System Block diagram

- A. Terminal Mobile phone.
- B. The peer Mobile phone fixed in the home.
- C. GSM communication between the peer Mobile phones (SMS & MMS).
- D. The hardware controlling system (microcontroller).
- E. The Computer device (PC).
- F. Sensors and cameras .

In the following paragraphs we will clarify system components usage and we will express how it works together in order to achieve system goals, and in the next chapter a more explanation will be presented.

Our system consists of many modules connected to each other. System can be divided into three main parts:

3.4 Sensing Unit

Sensors play an important role in satisfying our objectives to know about our surroundings. The initial measurements were just comparative and did not have a scale. Sensors have a long way from that point and we now have highly accurate measuring devices.

To form a complete application, however, use of these stand-alone sensors is not enough. Sensors must be interfaced to a microcontroller to form a complete system. Interfacing sensors with microcontrollers has revolutionized different applications and is continuing to do so. Many fields like medical, industrial, and automotive are benefiting from the latest innovations in sensor technology.

The use of sensors detects the conditions of the home (motion, smoke, fire, and door or window opening , etc) and the data is collected and transferred using a microcontroller. Man/Women need to visit his/her home frequently to examine it condition, or put person to monitor home for guarded purposes. Using the proposed system, data can be sent to personal mobile, allowing continuous monitoring of the home. For the purpose of this project, temperature, motion, smoke, and door, window sensors are addressed as shown in figure 3.2 Sensor that addressed used to interface with a microcontroller. These sensors can monitor the condition of the

home and, based on the thresholds chosen, record useful data and/or trigger alarms that activate other part in system to start their work

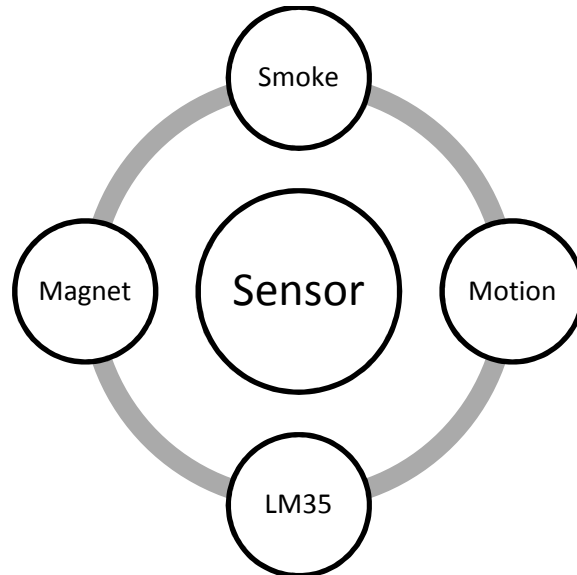


Figure 3.2 Distributed Home sensors

Sensor connected to PIC18F4550 to send signal if event occur to PC , in next section how to interface between sensor and PIC in part, and microcontroller to PC in other part.

3.4.1 Sensor to Microcontroller to PC.

The microcontroller receives information from the sensors connected to it by wires then transmit it to PC . In our project we will use serial port to allow communication between microcontroller and PC as shown in figure 3.3.

The microcontroller makes some analysis on the signals taken by sensor and passes them to computer, which may turn camera on.

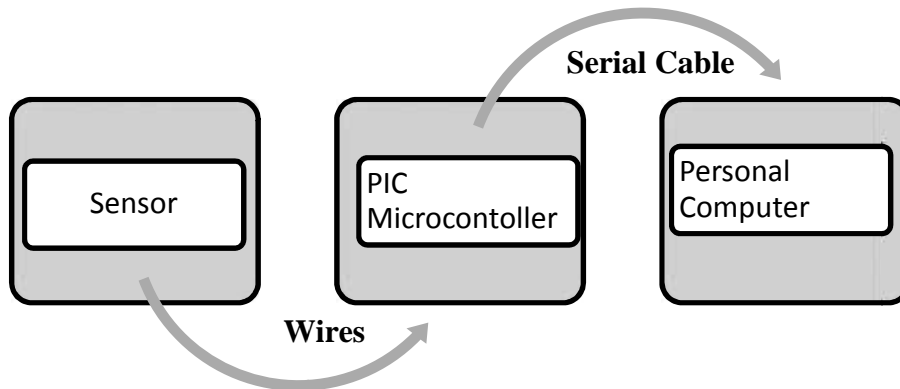


Figure 3.3 Sensors to microcontrollers to PC block diagram

3.5 Monitoring unit

3.5.2.1 Camera

Camera is the primary monitoring element; it is responsible for sending stream of video to the computer. To choose a specific camera we should take in Consideration some features including zoom, mega pixels.

Another important element for choosing camera is the connection type, where the camera can be connected via USB or other computer port.

In our system we choose a PC web cam (**TEAC Web Camera**) , as shown in Figure 3.4, to get continues stream which will framed in the system.

TEAC Webcam specifications

- ❖ Definition: 5.0 megapixels (2560x2048)
- ❖ CMOS chip type: Color CMOS Image Sensor
- ❖ High resolution: 2560*2048

- ❖ Resolution: 2560*2048, 1600*1200, 1280*1024, 640*480, 320*240, 160*120.
- ❖ Video format: 24-bit RGB
- ❖ Interface: USB1.1/2.0
- ❖ Frame rate: 320x240 up to 30 frames/second
- ❖ 1600*1200 up to 15 frames/second
- ❖ Sensor size: 1/6 inch
- ❖ S/N ratio: < 48 dB
- ❖ Built-in image compression
- ❖ Automatic white balance
- ❖ Automatic color compensation



Figure 3.4 TEAC Webcam

Camera Interface:

We have in our system two camera devices which are connected directly to USB port as shown in the following block Diagram.



Figure 3.5 cameras to PC Block diagram

How we programmed the camera?

Webcam Capture is a simple library for capturing images and video from a Webcam. It works using the Video for Windows API. It allows for both showing a video preview and simply taking pictures

Avicap32.dll is an Audio-Video Interleaved (AVI) Video Capture dynamic link library registered under Microsoft Corporation. It is developed by Microsoft to be part of its Windows operating system.

You can easily incorporate video capture capabilities into your application by using the AVI capture window class. AVICap provides applications with a simple, message-based interface to access video and waveform-audio acquisition hardware and to control the process of streaming video capture to disk. AVICap supports streaming video capture and single-frame capture in real-time.

A very common function of the avicap32.dll module is to enable Windows to capture movies and video images from cameras like digital cameras, web cameras, video cards and other video hardware. Furthermore, the avicap32.dll application stores the resulting video as AVI format. Once you have the DLL loaded you can create a capture window. A capture window you create by using the AVICap window class can perform the following tasks :

- Capture audio and video streams to an audio-video interleaved (AVI) file.
- Connect and disconnect video and audio input devices dynamically.
- View a live incoming video signal.
- Specify a file to use when capturing and copy the contents of the capture file to another file
- Set the capture rate
- Display dialog boxes that control the video source and format

The AVI format is the most common format defined by Windows for audio and video data on a computer. Basically, when a user records a video or is using a webcam for video conferencing, the video capturing process takes place.

The file `avicap32.dll`, described as AVI Capture Windows Class DLL, can be exploited further so that on-demand screen captures can be integrated into the .NET Windows application. In this enhanced use of the `avicap32.dll` module, you will be able to preview video input within the Windows application from your webcam. You can also record streaming video and capture images using the same video capture device.

In order to achieve this, the software Visual Studio .NET of Microsoft can be used. In this software, the module `avicap32.dll` constants are being declared in the video capture application.

3.6 Mobile Unit

In figure 3.6 show main component in mobile part .



Figure 3.6 Mobile Block Diagram

3.6.1 Terminal Mobile

The terminal mobile is carried by the owner and used to monitor and remotely control the system through GSM network.

Terminal mobile receives MMS message from Peer mobile, this MMS will tell the owner what happened in his home by using specific control word written as text inside message also it will receive an image to provide the owner with additional information about the events happened inside the home. as shown in Figure 3.6

3.6.2 Peer mobile

Mobile in the home will connect to a PC, when event happen PC will send information to this mobile which reshape this data as text and receive image captured by camera and send it plus text as MMS message to the terminal mobile carried by house owner.

3.6.3 Interface pc to peer mobile

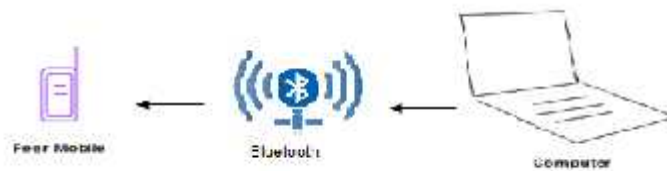


Figure 3.7 Mobile to PC interface

As shown in figure 3.7 , we use Bluetooth to send image from PC to peer mobile. Firstly, we create a connection channel to peer mobile then send image and text that describe events occurs.

3.6.4 How to deal with MMS

MMS is a big new world, and it's all too easy to get lost out there. We decide to use MMS in our system in addition of text in order to give user a clear vision about the event. Now we would like to provide a good understanding of the basics of MMS.

MMS message is like SMS message ,most people are familiar with at least the basics of sending an SMS message. Figure 3.8 show how an SMS message is delivered. There are several bits missing, but the basic concept is:

- The message originator(Peer Mobile) addresses the short message to the receiver.
- The phone contain information about SMSC (SMS Center) ,and the message is sent there .
- SMSC is attempt to forward the message to the receiver.



Figure 3.8 Sending an SMS message

The basic concept of sending an MMS message is exactly the same as shown in figure 3.9:

- The message originator (Peer Mobile) addresses the multimedia message to the receiver.
- The phone contain information about MMSC (MMS Center) ,and the message is sent there .
- SMSC is attempt to forward the message to the receiver.

If for some reason the receiver is unreachable, MMSC stores the message for a time, and if possible, delivers the message later. If the message cannot be delivered within a certain time frame, it is eventually discarded.

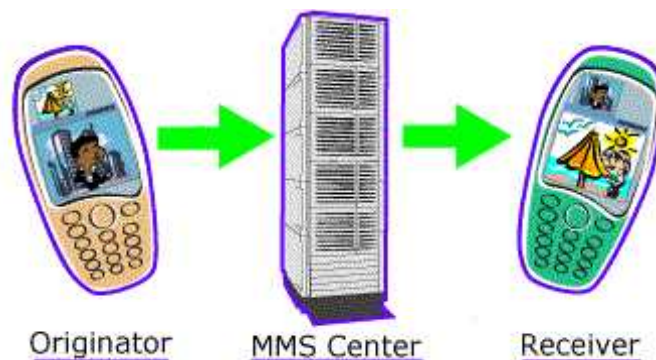


Figure 3.9 Sending an MMS message

Is it really that simple? No, in fact it is more complicated, figure 3.8 is missing much more, and here is the more detailed version of MMS sending:

The message originator addresses it to the receiver, the phone contains information about MMSC and initiates a WAP connection and sends the message, MMSC accepts the message and responds to the originator over the same WAP connection and the originator phone indicates " Message Sent ".

Then MMSC attempts to send an indication message to the receiver, the receiver phone must be set to accept the MMS so it initiates a WAP connection and retrieves the MMS from the MMSC and the receiver phone indicates " Message Received ". The receiver phone acknowledges receipt, still over the same connection. MMSC indicates the originator that the message was delivered, the originator phone indicates " Message Delivered ".

3.7 How system works

Now, we will explain how system components work together in order to achieve project goals.

Owner has the capability of turning the system On/Off whenever he wants. While the system is running, if any abnormal events happened such as fire, motion; sensors will detect it and send signal to microcontroller.

After that, PIC makes some processing of received signal, converts it to digital signal, and then transmits it to PC serially.

PC, after receiving signal serially makes some local analysis in order to turn event's room cameras on .Then camera will captures a stream of images and videos that will be transmit to PC serially using USB cable. PC then store captured images and videos in a file for later owner usage.

After that an image will send to peer mobile through Bluetooth in addition to text useful information about system state.

Peer mobile prepare MMS message containing event image ,also containing a text about house state information, then send to Terminal Mobile after establish GSM communication.

Terminal mobile which always carried by the owner, it receives MMS message from Peer mobile; which means owner alarming.

This MMS will tell the owner what happened in his home by using specific control word written in the MMS in addition to an image provide the owner with event nature occur inside the home.

Here the following step describes that:

- Initially, our system is in ready state and waiting for event occurrence.
- Sensing unit detect any abnormal events occur (motion, smoke, temperature and window/door event).
- Sensing unit send predefined word to PC that defined natures of event occur.
- PC triggers cameras for capture images and record video.
- PC builds Bluetooth Connection with peer mobile.
- PC send an image and text files to peer mobile.
- Peer mobile will receive image and text form PC.
- Peer mobile build MMS message and send it to terminal mobile.

3.8 System Flow chart

Figure 3.10, shows the system flow chart, it helps to understand and analyze the steps and the problems might appear and needs to be solved, in order to make the system as flawless as possible:

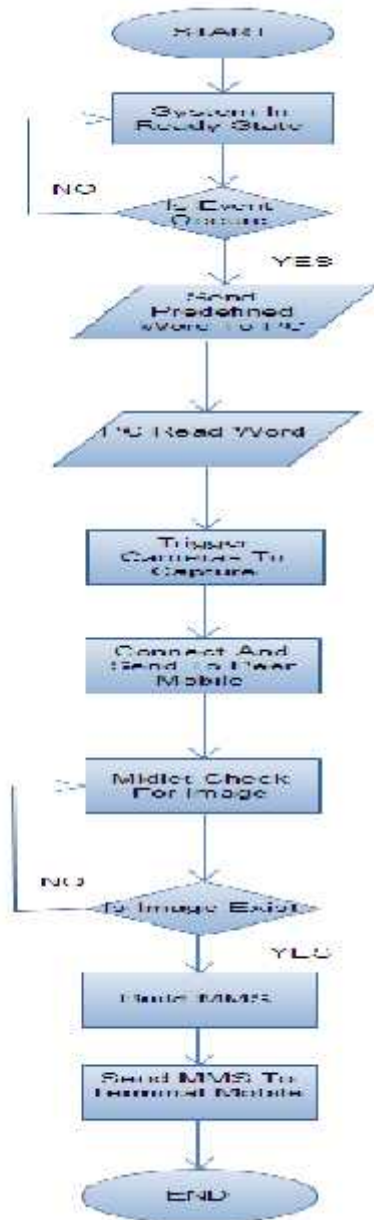


Figure 3.10 System Flow Chart

4.1 Introduction

As we shown in previous chapter, our system has mainly three parts, sensing unit, monitoring unit and mobile unit part. A division must be made between these parts in order to build the system.

This chapter describes designing the whole system; section 4.2 describes sensing unit design; section 4.3 describes monitoring unit and section 4.4 describes mobile unit design.

In figure 4.1, a general system interfacing block diagram is shown, it represents the interfacing of the whole system. In the coming sections, detailed interfacing techniques and requirements will be discussed.

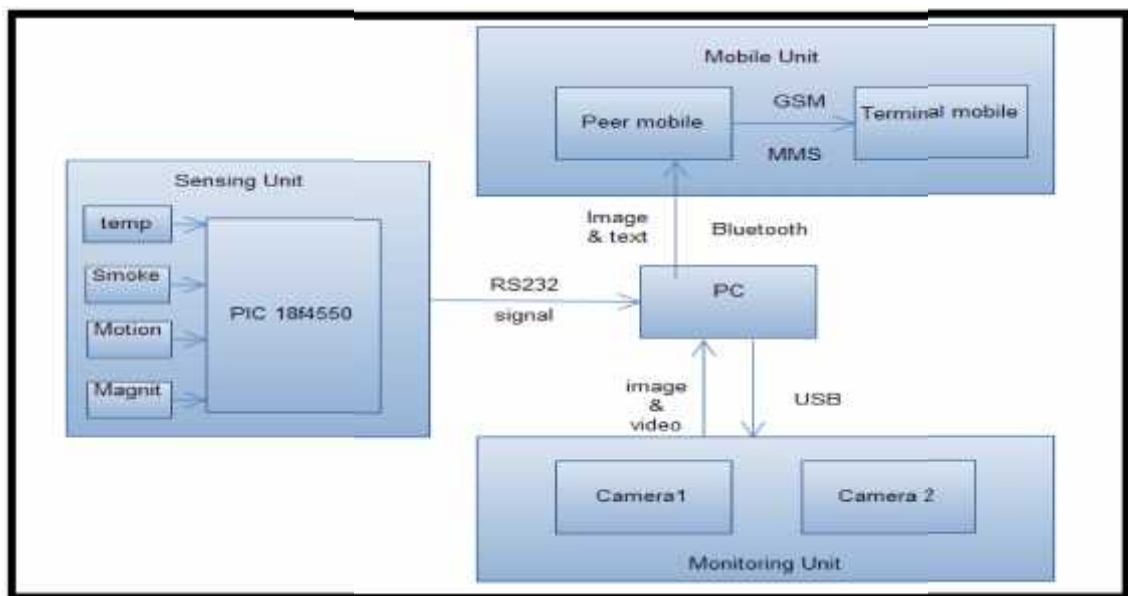


Figure 4.1 General System Interfacing Block Diagram.

We will be explain component of figure 4.1 in the next sections

4.2 Sensing Unit

Sensing unit in chapter four describe how we can connected in detail between sensor and PIC microcontroller ,and some important thing should be considered before make interfaces between sensor as will explained in next paragraph

4.2.1 Requirement must find in microcontroller :

First to interface sensor with PIC 18F4550 ,we should need to know about the minimum requirement for PIC to run .

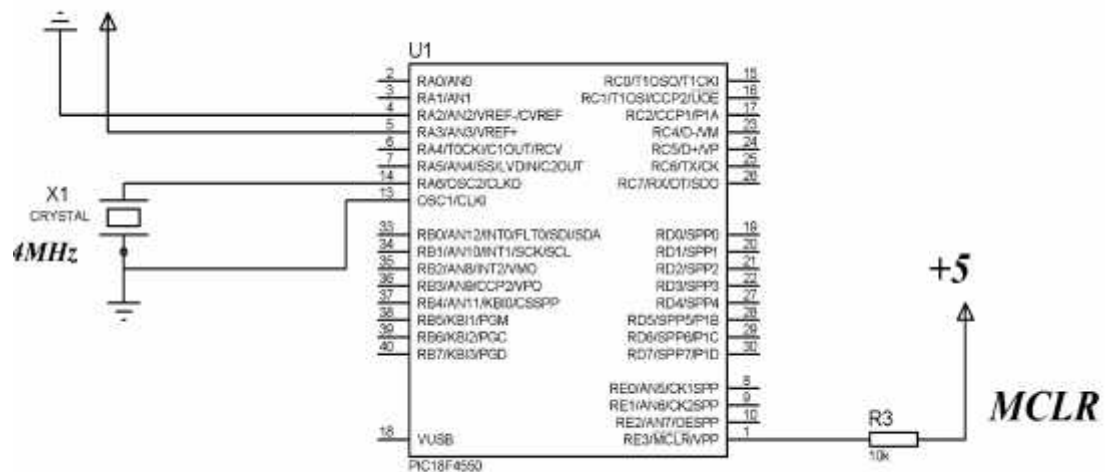


Figure 4.2 Minimum Requirements for PIC to Run

Three main thing in PIC should be connected as shown in previous figure :

- 1- Vref- (0V) – Vref+ (5V) .
- 2- Crystal at pin 13,14 consequently .
- 3- MCLR .

One is explained itself ,2 and 3 will be explain next:

Master Clear Reset (MCLR)

The MCLR pin provides a method for triggering an external Reset of the device. A Reset is generated by holding the pin low. These devices have a noise filter in the MCLR Reset path which detects and ignores small pulses. The MCLR pin is not driven low by any internal Resets, including the WDT. *In PIC18F4550 devices ,the MCLR input can be connected to resistor then to 5 volts.*

Important peripheral should be explain that used in project :

- 1- Oscillator frequency .
- 2- Watch dog timer.
- 3- Low voltage detect.
- 4- Analog to digital converter .
- 5- Serial port ,USART.

Oscillator frequency

The operation of the oscillator in PIC18F4550 devices is controlled registers. Configuration registers, select the oscillator mode. As Configuration bits, these are set when the device is programmed and left in that configuration until the device is reprogrammed. The *OSCCON* register selects the Active Clock mode; it is primarily used in controlling clock switching in power-managed modes.

Oscillator Types

PIC18F4550 devices can be operated in twelve distinct oscillator modes. From these mode we use external oscillator (HS) that have 4MHz value.

In HS Oscillator mode, a crystal is connected to the OSC1 and OSC2 pins to establish oscillation. Following figure shows the pin connections.

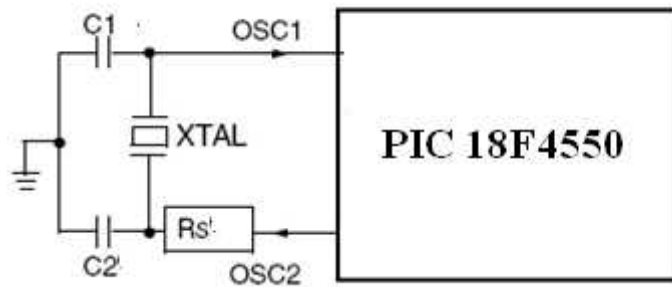


Figure 4.3 Crystal Operation .

The watchdog timer

It is a free running down counter which , ,if enabled and allowed to time out ,will cause the microcontroller to reset ,so in our project should make it OFF .

Low voltage detect

The ability to detect a failing power supply is very valuable in an embedded system . In a battery –powered product this can be used to detect a failing battery . It can also be used to detect power being switched off . In either case , the microcontroller may wish to activate a warning signal or exercises an orderly shutdown , perhaps saving key operating variable to EEPROM . In the case of power loss. also this peripherals should make it OFF in our project.

Analog to digital converter:

In chapter two take about ADC , which convert analog signal to digital that can be read by computer .The following steps should be followed to perform an A/D conversion:

1. Configure the A/D module:

- Configure analog pins, voltage reference and digital I/O (ADCON1)
- Select A/D input channel
- Select A/D acquisition time
- Select A/D conversion clock
- Turn on A/D module

2. Configure A/D interrupt (if desired)

3. Wait the required acquisition time (if required).

4. Start conversion:

5. Wait for A/D conversion to complete (interrupt ,polling).

6. Read A/D Result.

7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 3 TAD is required before the next acquisition starts.

Serial ports

PIC 18F4550 has two major serial modules :the master synchronous serial port and Universal Synchronous Asynchronous Receiver Transmitter (USART).

Calculation needed (*Note : equation find in chapter two in USART part of serial port section*)

✱ 8MHz oscillator

✱ 9600 baud asynchronous

✱ For BRGH = 1

$$SPBRG = 4000000/(16 \times 9600) - 1 = 51.08$$

✱ For BRGH = 0

$$SPBRG = 4000000/(64 \times 9600) - 1 = 12.02$$

✳ Best choice is BRGH = 0, SPBRG = 12.02

As of a baud rate calculation, the microcontroller operating at 8MHz and we use 9600 baud to communicate with a serial port on a PC. USART used in asynchronous mode. We choice BRGH =0 , because less error than BRGH=1 .

Note that to get an accurate and stable baud rate, an accurate and stable oscillator is required.

4.2.2 Sensor Circuit

In chapter two we take about main sensor that used to detect event on our home , in this chapter we talk about sensor circuit , interfaces with PIC microcontroller ,and needed programming for each sensor to read value of it.

Used sensor in our project divide in two part as show in figure 4.4 : analog sensor and digital sensor . analog have special configuration to work , but digital one work as digital switch.

- 1- LM35 temperature sensor that used to indicate fire occur in home.
- 2- Motion detector to sense any motion (persons only) .

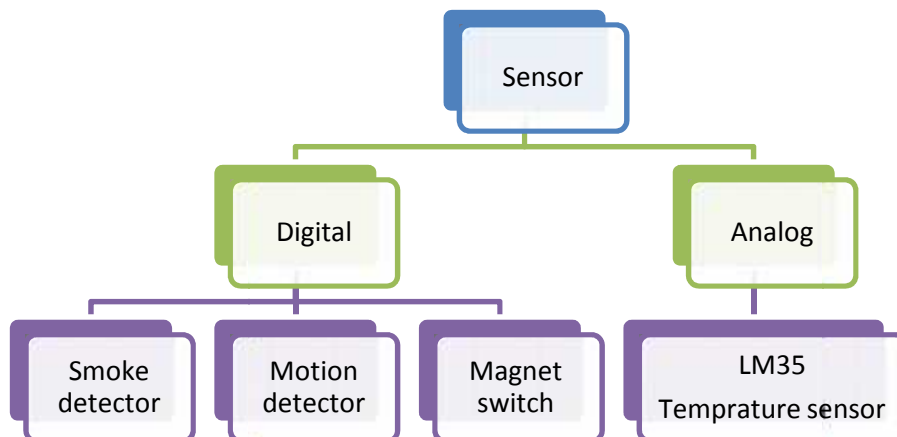


Figure4.4 type of sensors

3- Smoke detector to sense if any smoke in room.

4- To increase reliability of our system use Magnate switch for detecting door or window opening .

LM35 (Precision Centigrade Temperature Sensors):

From one of the main objective on our system to monitoring temperature , usually ,a temperature sensor converts the temperature into an equivalent voltage output . IC LM35 is a such sensor .

Figure 4.5 shows the functional block diagram of PIC18F4550 based temperature monitoring system.

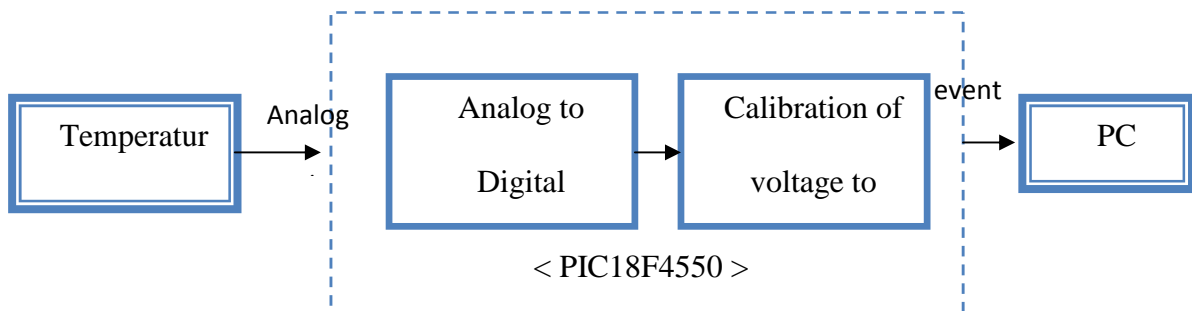


Figure 4.5 block diagram of LM35.

In block diagram show that the sensor measure temperature and because the sensor value is analogue and the microcontroller read only digital value so we need analog to digital converter , we use built in A/D converter.

Hardware description

LM35 have 3 pins ,connect pin1 to ground ,pin2 (output of sensor) to PORTA of PIC , and pin3 to Vcc (5 volt). The output of the sensor is fed to the internal ADC

of the microcontroller.(RA0/AN0 is channel-1 of the internal ADC) . The analogue voltage output of the sensor is converted into its equivalent digital value by the ADC and then its equivalent degree Celsius value is calculated by the software. the calculated value act as determinant that event occur or not.

Before start talk about software description , we describe how to calculate the value of the sensor .The output linearly varies with temperature. The output is 10MilliVolts per degree centigrade. So if the output is 310 mV then temperature is 31 degree C. The internal ADC of the microcontroller has 13 channel of analogue input and gives 10-bit digital output. In this project , the reference voltage to the ADC is the same as the supply voltage to the microcontroller ,i.e,5V . The resolution of the ADC can be calculated as follows :

$$\text{Resolution} = \frac{V_{\text{ref}}}{1024-1} \text{mV} \dots\dots\dots \text{equation 4.1}$$

as it a 10-bit ADC = 5/1023 = 4.887 . It mean that for 4.887 change in analogue input ,the ADC output change by binary "1" with a reference voltage of the 5V.

So the voltage output (in volts) of the sensor is :

$$\frac{\text{ADC result} \cdot 5}{1023} \rightarrow \text{The temperature in degree Celsius is :}$$

$$\frac{\text{Sensor output} \cdot 1000}{10}$$

1000 because sensor output in mV , and 10 because sensor represent for every 1 Celsius degree 10mV.

The last thing should explain is AD configuration:

We are running at 4MHz in our project so we set prescaler of 8 TOSC.

Our F_{OSC} = 8MHz

Therefore our T_{OSC} = 1/8MHz
= 125nS

$$\begin{aligned}
16 T_{OSC} &= 16 \times 125 \text{ nS} \\
&= 2000 \text{ nS} \\
&= 2 \text{ uS}
\end{aligned}$$

2 uS is more than the minimum requirement.

now we have the T_{AD} we can calculate the division factor for acquisition time. Acquisition time can be specified in terms of T_{AD} . It can be set to one of the following values.

- 20 x T_{AD}
- 16 x T_{AD}
- 12 x T_{AD}
- 8 x T_{AD}
- 6 x T_{AD}
- 4 x T_{AD}
- 2 x T_{AD}
- 0 x T_{AD}

The safe acquisition time is 2.45uS, so we select 2 x T_{AD} as acquisition time.

$$\begin{aligned}
T_{ACQ} &= 2 \times T_{AD} \\
&= 2 \times 2\text{uS (Replacing } T_{AD}= 2 \text{ uS)} \\
&= 4\text{uS}
\end{aligned}$$

4 uS is more than required 2.45uS so it's ok.

Figure 4.6 show connect sensor to our PIC as explained previously .

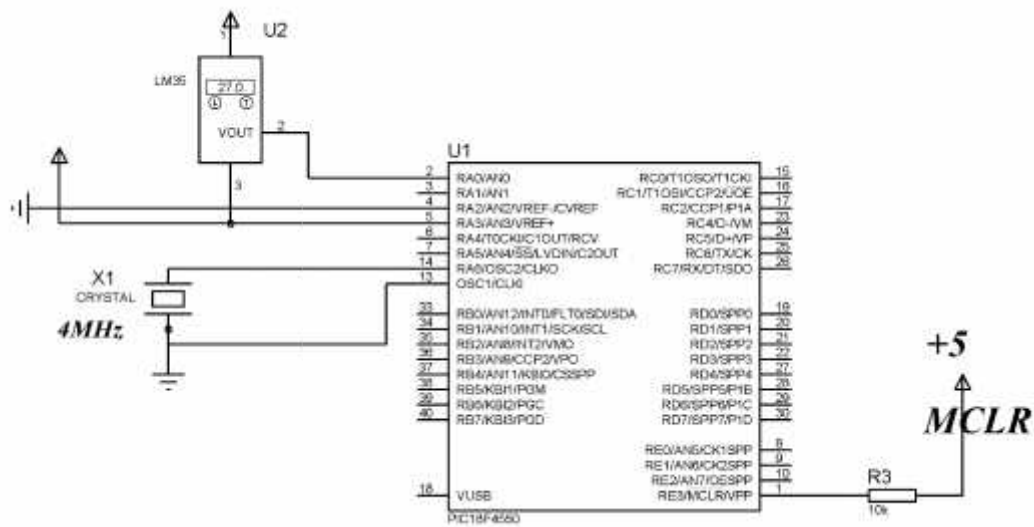


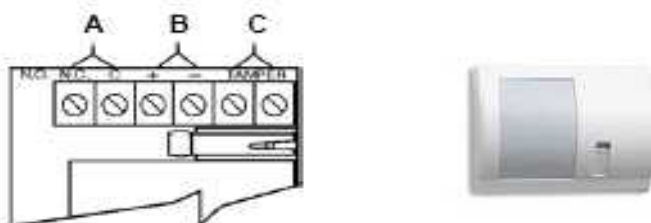
Figure 4.6 interface LM35 sensor to PIC18F4550

Temperature threshold :

We use temp = 35.0 and over to indicate a fire occur and high temperature in home.

Motion detector :

Device that indicate if dangerous movement in home or not. This sensor have special way to connected it to our circuit , first we remove front cover of the detector , to show these pins A ,B , and C as shown in this figure



A not connected because we don't need it, connect two wires to B (negative to negative side, make the same as positive), C pins connect to another two wires. In data sheet of the detector table Technical specification, show that the detector works at 12 V, and our PIC runs only at 5V as maximum so we need a transformer to give suitable voltage. B connected to transformer, C connected to circuit as we connect a switch so when any foreign enters the home it goes low to indicate a danger event. The following figure shows the interface between LM35 and motion to PIC.

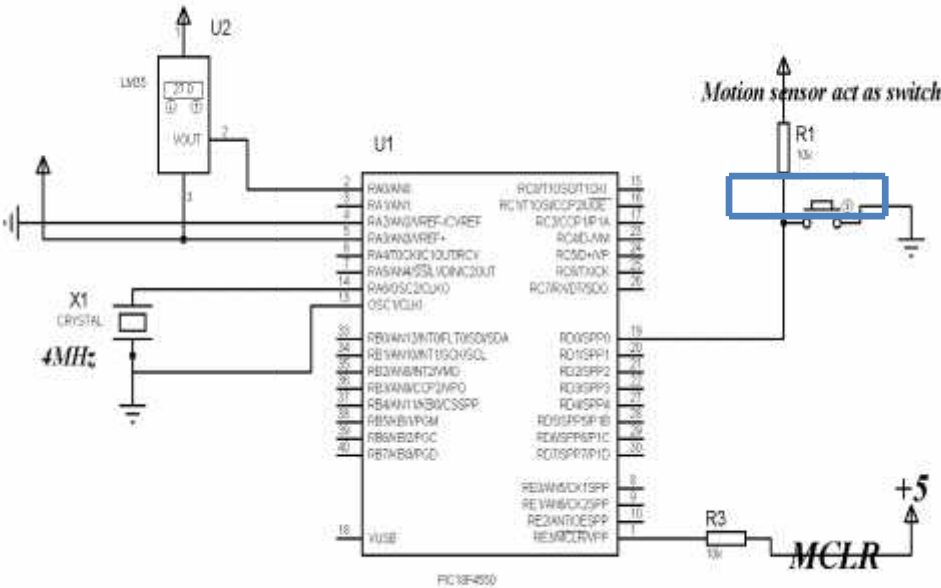


Figure 4.7 Interface LM35, Motion to PIC 18F4550

Location of motion detector :

Position the detector so that it faces the area you want to illuminate. To test the range, walk in front of the sensor's detection area at the farthest point you want the sensor to reach.

Smoke sensor :

Its nearly connected to circuit as motion detector as shown in the following figure .

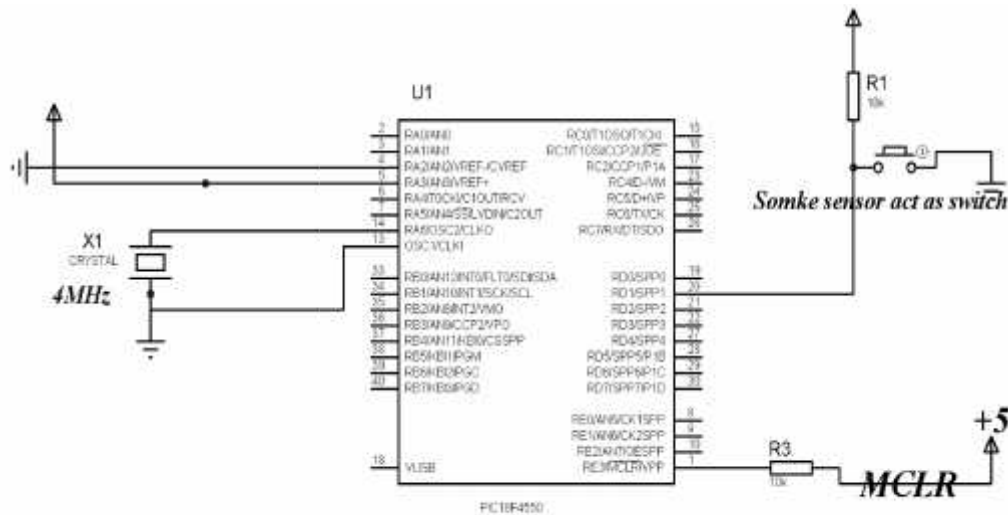


Figure 4.8 Interface Smoke to PIC

How to Place a Smoke Detector ?

It is important to properly place the smoke detectors in the home. In most locations throughout the U.S., smoke detectors are required in all new homes. The number and location are listed below:

Number and Location Requirements

The minimum standard as stated in the National Fire Prevention Association's National Fire Alarm Code (NFPA 72): There should be a smoke detector on every level of the house, including the basement and outside every bedroom

New homes require hard-wired alarms to be interconnected so that if one alarm is activated, all alarms will sound the alarm signal.

New homes require smoke detectors in every bedroom. On floors without bedrooms, smoke alarms should be installed in or near living areas, such as family rooms and living rooms.

Placement Requirements

As stated by the NFPA: "Since smoke and deadly gases rise, alarms should be placed on the ceiling at least 4 inches from the nearest wall, or high on a wall, 4-12 inches from the ceiling. This 4-inch minimum is important to keep alarms out of possible "dead air" spaces, because hot air is turbulent and may bounce so much it misses spots near a surface. Installing alarms near a window, door or fireplace is not recommended because drafts could detour smoke away from the unit. In rooms where the ceiling has an extremely high point, such as in vaulted ceilings, mount the alarm at or near the ceiling's highest point."

Magnet switch :

Its use in door and window to increase security of our home . To use magnet it connect in the same method use in smoke and motion detector .

All schematic design :

As shown in figure 4.9 sensing unit contain pic18F4550 that connected to two type of sensor ; analog sensor and digital sensor , analog sensor (LM35) connected to pin 1 of port A and other pins of LM35 connected to power and ground .Digital sensor (smoke , magnet , motion) connected to port D , pin 1,2,3 respectively .

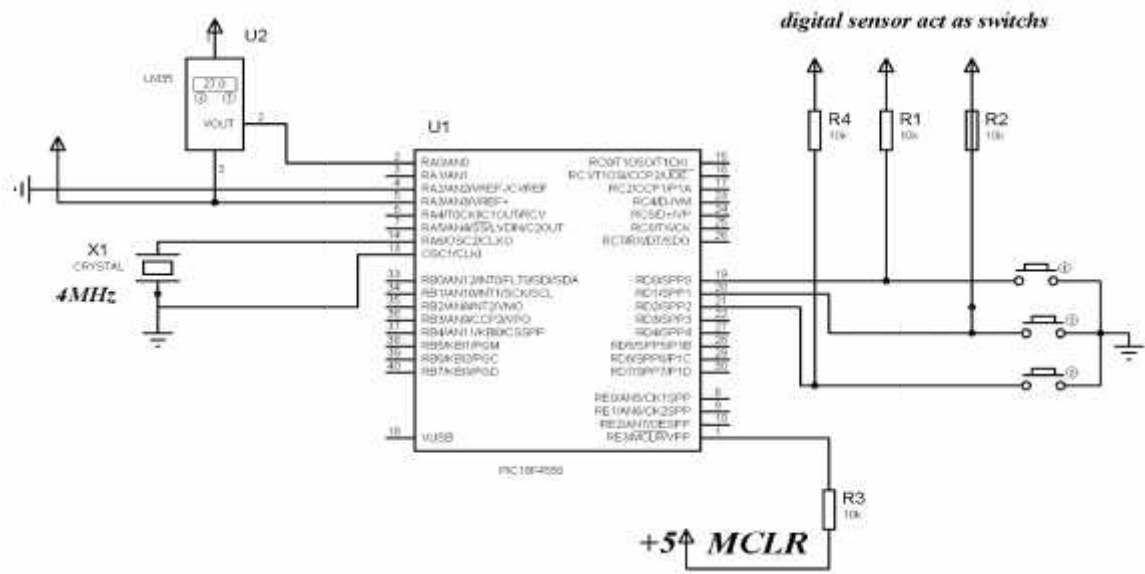


Figure 4.9 All system interface

4.3 Monitoring unit

4.3.1 Camera's functions

Get an image

The Target of the monitoring system is to provide images representing the current system state when events occur. So the main function is to capture an image and send it to PC where is to be saved as image file.

Record video and save it in video file

When user needs to initiate video recording service for later using. It will be saved into files, the same with other techniques and other types of cameras. We can also customize specific type of file to store video which can decrease the output video size especially when recording large blocks of data.

Function that are performed by home-owner:

1. Select camera to view.
2. Display camera views in a PC window.
3. Take picture and video from the camera.
4. Replay camera output.

4.3.2 camera's technique

While our system in ready state and all camera on, the system continuously read from serial port according to specific timer, after reading from serial port we have two states:

- Waiting state which means no event occur.
- Event state.

If we have *waiting* state we must still read again from serial port every period timer interval and we can get an image that show the state of home which save locally in PC . If we have *event* state, then, our system capture groups of image describe event nature from camera number one, and record video with specific period time (10 minute) from camera number two. All images and videos store locally in pc.

4.3.3 Interfacing between PC and peer mobile.

The first thing you should do is connect to the peer mobile. You can choose the peer mobile using its MAC address. After the peer mobile has been selected, we connect to it and initiate a new session, then we load image from specific location on PC and send it to peer mobile. Here we have state diagram that describe the transition between states.

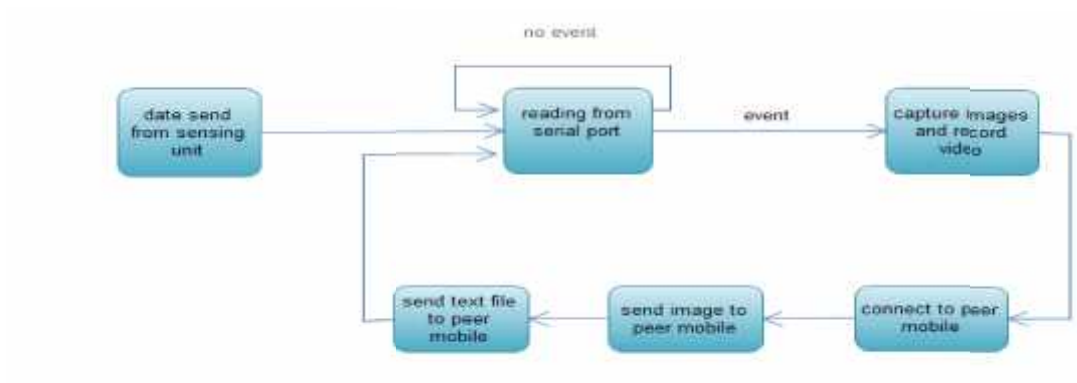


Figure 4.10 State Diagram For Monitoring Unit

4.4 Mobile unit

Communication between the Peer mobile and Terminal mobile is done through GSM technology. There is more than one option to communicate with the system through mobiles:

SMS

Which can be used to alert the system user through send a text message containing few words describe threat. Its good way but for us it is not enough so we used MMS.

MMS

Very important technology defined in many mobile phones we will used MMS in order to provide user with an image to show him exactly what happen in his home, as we all know it is more better to see something rather than hear about. Because of that we use the two technologies in one message, include text and image.

4.4.1 Monitoring Using Mobile Messages

As we say previously we will use both of SMS and MMS technology in our system. Multimedia Message Service is the ways that enable user to get his house information in our system which will be send by the peer mobile , MMS message consists of two parts , first is a text describe what had happen inside home ,this information is delivers to peer mobile from the sensors spreads inside home ,this text will be used instead of sending a separate SMS, the second part is an image of the event which give user his clear vision, this image also delivered to peer mobile from cameras inside home.

The Peer Mobile connected to PC directly and make continuous checking on its memory card through its J2ME application which run all the time ,as soon as the image file exist in memory this application will find it in approximately real time and also go to read text file which must be together with image file. This text and image will formed as our MMS message which send immediatly to house owner mobile ,thus we have alerted the user. Here the following state diagram will explain how this happen .

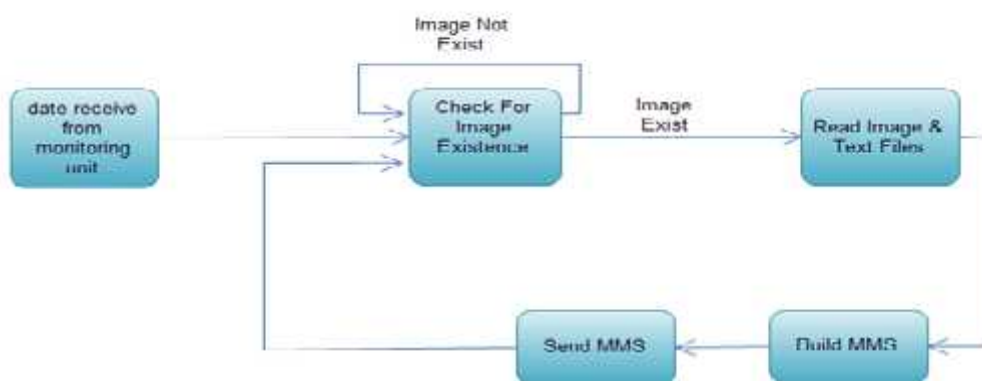


Figure 4. 11 State Diagram For Mobile Unit

4.4.2 J2ME Application

It is called MIDlet application, write once run anywhere. MIDP does not run in the “ regular ” Java fashion using Main() , System.exit() methods. Instead, we use MIDlet applications which are subclasses of: javax.microedition.midlet.MIDlet that is defined by MIDP. The MIDlet class defines abstract methods that the main class implements (for example: startApp(), destroyApp(), notifyDestroyed()) as shown in the following figure .

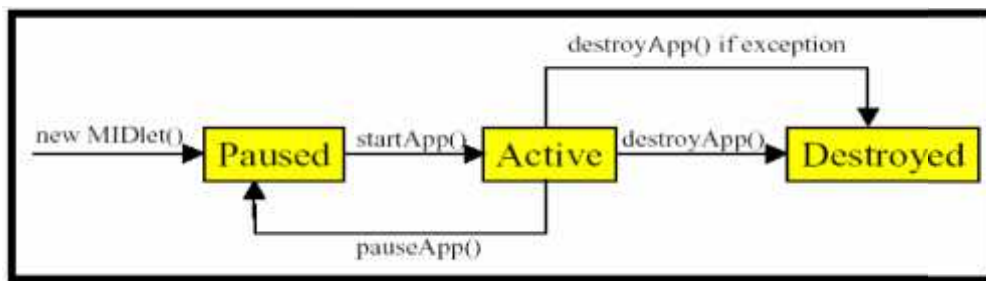


Figure 4.12 MIDlet Main Class Mmethods

MIDlet Suite

One or more MIDlets are packaged together into a MIDlet suite, composed of:
JAR (Java archive) file - The JAR file contains Java classes for each MIDlet in the suite and Java classes that are shared between MIDlets. The JAR file also contains resource files used by the MIDlets and a manifest file.

JAD (Java Application Descriptor) file - This file contains a predefined set of attributes that allows the device application management software to identify, retrieve, and install the MIDlets .

4.4.3 Mobile Specification

We choose Nokia 2690 mobile figure 4.13 , as peer mobile, which support MMS technology , and a SIM cards inside it to send messages, Also memory card inside peer mobile. It is classified as MIDP 2.0 mobile.



Figure 4.13 :Nokia 2690

It supports :

1. MMS technology.
2. Capture picture resolution of 480 * 640 pixels.
3. pictures size until 1600 * 1200 pixels.
4. Java application specially J2me application.
5. Bluetooth technology .

5.1 Introduction

In this chapter, the detailed description for the project software, in previous chapter we take about how can interface between sensor and microcontroller from hardware description, in this chapter we will describe software used to give hardware interfaces goal.

Then we will be explaining how we connect cameras to our system and the method that used to program our video device in order to capture picture and record video. And we will be offer how to send an image and text to peer mobile.

Also this chapter describes Mobile Programming in details, these is a very interested side in our project and open to us a completely new window to programming world, our mobile programmed using J2ME language in order to create the JAR file which used to install on mobile to run the application on it.

5.2 Software Requirement Specification

5.2.1 Software description of sensor interfaces

The software code written in 'C' language and compiled using C18 cross compiler in MPLAB IDE .Code handle event occur from analog sensor and digital sensor , then it handle send event serially to PC through USART (configure serial port).

For analog sensor

- 1- Initiate analog –to – digital conversion and obtain the result.
- 2- Calculate the temperature in degree Celsius from the voltage value.
- 3- Use this value to determine if the events occur or not.

For digital sensor

- 1- Test pins of port if it trigger or not.
- 2- Take decision depend on status of pins.

After that we inform the system that event occur by using predefined control word.

MPLAB:

MPLAB IDE is a software program that runs on your PC to provide a development environment for your embedded microcontroller design.

The design cycle for developing an our project is

- 1) Determine design based on the associated hardware circuitry.

- 2) Knowing which peripherals and pins control hardware, write the software. We use a compiler that allows a more natural language for creating programs. With these Language Tools we can write and edit code that is more or less understandable, with constructs that help organize our code.

- 3) Compile or assemble the software using a Language Tool to convert code into machine code for the PICMicro device. This machine code will eventually become firmware, the code programmed into the microcontroller.

- 4) Test code. Usually a complex program does not work exactly the way might have imagined, and “bugs” need to be removed from design to get it to act properly.

- 5) “Burn” code into a microcontroller and verify that it executes correctly in finished application.

MPLAB environment as show in figure5.1 first it contain untitled workspace which contain needed file after create project ,every needed file contain main four part Header file (p18F4550.h) ,Linker file (p18f4550_g.lkr), library file (p18f4550.lib) ,source file → add file that contain code of project . Output window for sure that code correctly executes.

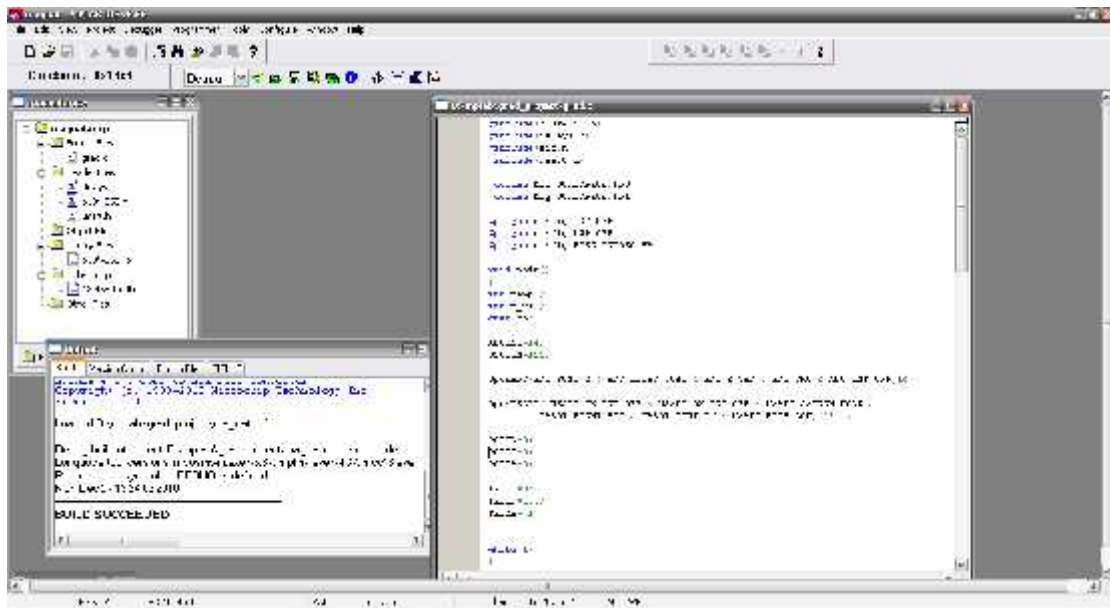


Figure 5.1 MPLAB Environment

After complete the code, can we use memory usage gauge to know how many program take capacity as shown in following figure:

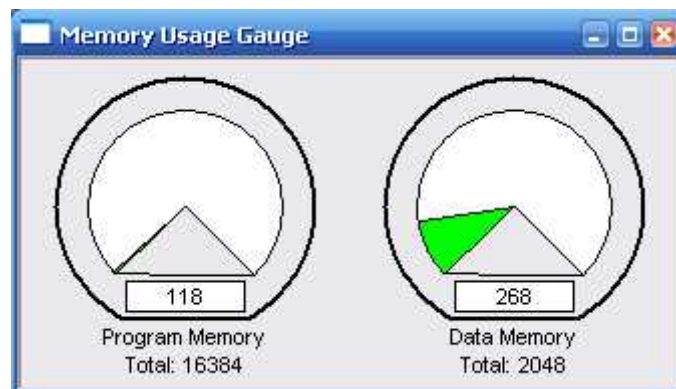


Figure 5.2 Memory Usage Gauge

To download program in PIC can use one of programmer reference in MPLAB, we use PIC KIT 2.

PIC KIT 2

- In-Circuit-Debugging with MPLAB® IDE
- Debug in the application circuit
- Supported Device Families: PIC10F, PIC12F, PIC16F, PIC18F, and PIC24.

Now we should explained code that test sensor status and send char to PC , before that we mention to important point that we have four sensor and byte contain 8 bits , so assume that every sensor represent one bit we need 4 bit to represent the status ,other 4 bit take as don't care and equal 0.

Let bit 0 represent motion , bit 1 represent smoke , bit 2 represent temp , bit 3 represent magnet ,temp > 35 → 1, temp < 35→0 .

Mag	Temp	Smk	Mot	Char	Status
0	0	0	1	A	Motion detecting
0	0	1	0	B	Smoke detecting
0	0	1	1	C	Motion , Smoke detecting
0	1	0	0	D	Temp high→over threshold
0	1	0	1	E	Temp high , Motion detecting
0	1	1	0	F	Temp high , Smoke detecting
0	1	1	1	G	Temp high , Motion ,Smoke detecting
1	0	0	0	H	Magnet trigger → open door
1	0	0	1	I	Magnet trigger , Motion detecting
1	0	1	0	J	Magnet trigger , Smoke detecting
1	0	1	1	K	Magnet trigger , Motion detecting
1	1	0	0	L	Magnet trigger, Temp high
1	1	0	1	M	Magnet trigger , Motion detecting, Temp high
1	1	1	0	N	Magnet trigger , Temp high , Smoke detecting
1	1	1	1	O	All sensor make event

Code will be explained in appendices but we explain main code function. In our code we use main two libraries that c18 provided which is analog to digital converter library, and USART library as in following figure respectively:

Analog to digital converter library can be found on ADC.h

Following figure show the main function used in ADC



Figure 5.3 Step A/D Conversions

OpenADC

ADC_RIGHT_JUST → we have two register store value of conversion so we can determine if read left justified or else. (Result in Least Significant bits)

ADC_2_TAD → 2 Tad

ADC_CH0 → Channel 0 (AN0)

ADC_INT_OFF → Interrupts disabled

USART library:

Main step for USART using c18 library as shown in figure 5.4

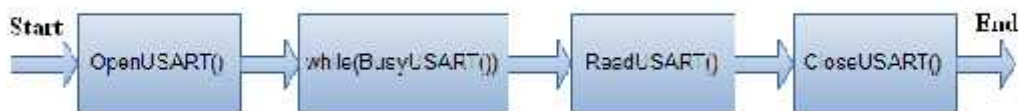


Figure 5.4 USART Library

OpenUSART include :

USART_TX_INT_OFF: Transmit interrupt OFF

USART_RX_INT_OFF: Receive interrupt OFF

USART_ASYNC_MODE: Asynchronous Mode

USART_EIGHT_BIT: 8-bit transmit/receive

USART_BRGH_HIGH: High baud rate

12: value calculated for register spbrg

The **flowchart** that describe code is

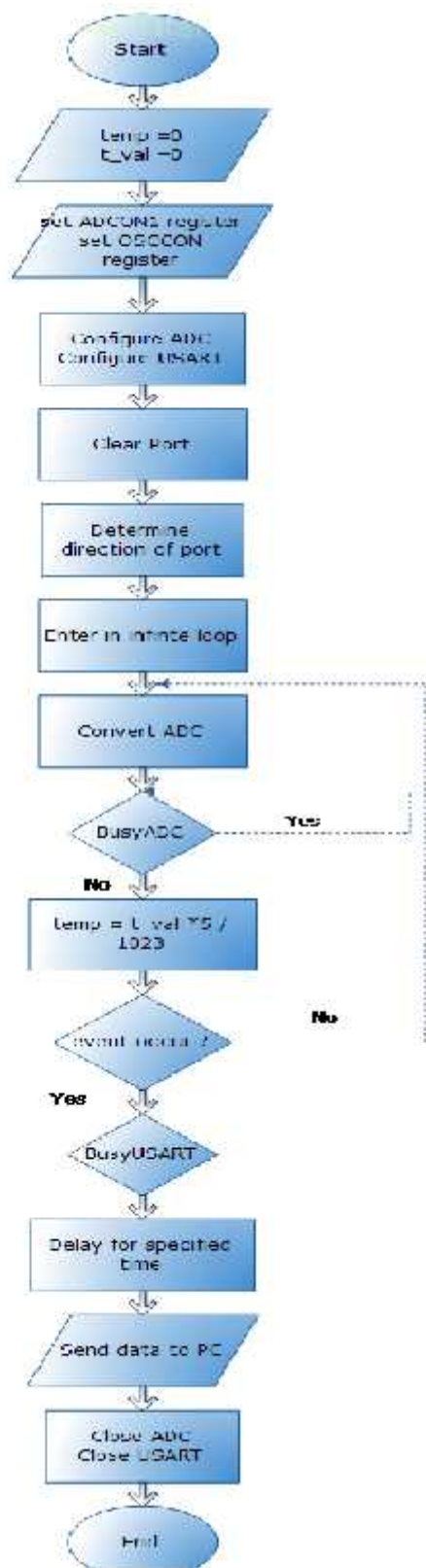


Figure 5.5 Flow chart of sensing unit

5.3 Software Description of Cameras

We use visual basic .NET for programming and connect to video devices; initially the video devices are connected to USB port as shown in the following figure.



Figure 5.6 USB Webcam

Here we will explain some steps to connect video devices to our program

1. Creating a Capture Window

A capture window interfaces with the video capture driver and displays a preview of the video data.

```
Declare Function capCreateCaptureWindowA Lib "avicap32.dll"  
( ) As Integer
```

2. Send Message Function

Sends the specified message to a window or windows. The Send Message function calls the window procedure for the specified window and does not return until the window procedure has processed the message.

```
Declare Function SendMessage Lib "user32" Alias  
"SendMessageA" (ByVal hWnd As Integer, ByVal wParam As  
Integer, ByVal lParam As String) As Integer
```

This function send message (wMsg) to specific window (hWnd).

3. Video Capture Messages

```
Const WM_CAP_START = &H400S
Const WM_CAP_DRIVER_CONNECT = WM_CAP_START + 10
Const WM_CAP_DRIVER_DISCONNECT = WM_CAP_START + 11
Const WM_CAP_SET_PREVIEW = WM_CAP_START + 50
Const WM_CAP_SET_PREVIEWRATE = WM_CAP_START + 52
Const WM_CAP_STOP = WM_CAP_START + 68
Const WM_CAP_ABORT = WM_CAP_START + 69
Const WM_CAP_SEQUENCE = WM_CAP_START + 62
Const WM_CAP_SET_SEQUENCE_SETUP = WM_CAP_START + 64
Const WM_CAP_FILE_SET_CAPTURE_FILE = WM_CAP_START + 20
```

Right, so you have a capture window. At this stage the capture window appears as a black rectangle in your program. Before it can capture video you need to connect it to a device using WM_CAP_DRIVER_CONNECT message .

The WM_CAP_DRIVER_DISCONNECT message disconnects a capture driver from a capture window.

The WM_CAP_SET_PREVIEW message is used to enable and disable preview mode. The -1 in this call enables preview mode, a 0 in its place is used to disable preview mode.

The WM_CAP_SET_PREVIEWRATE message sets the frame rate. In our program we set the frame rate to 30 frames per second.

The WM_CAP_ABORT message stops the capture operation. In the case of step capture, the image data collected up to the point of the WM_CAP_ABORT message will be retained in the capture file

The WM_CAP_STOP message stops the capture operation.

The WM_CAP_FILE_SET_CAPTURE_FILE message names the file used for video capture.

The WM_CAP_SEQUENCE message initiates streaming video and audio capture to a file.

4. Sample Code for Display Video from Webcam

```
WebCam webcam = new WebCam();  
  
// passing Image Control Element  
webcam.InitializeWebCam(imgVideo1)  
  
// start webcam video playing  
webcam.Start()  
  
//stop video playing  
webcam.Start()
```

For Show Resolution Setting and Video Format Dialog

```
Webcam.config();
```

For Show Advance Setting Dialog

```
Webcam.config2();
```

The following figure show video format dialog and video source dialog

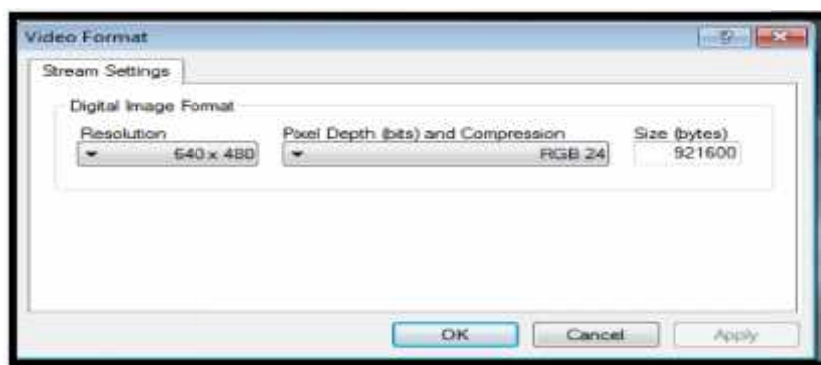


Figure 5.7 Video format dialog

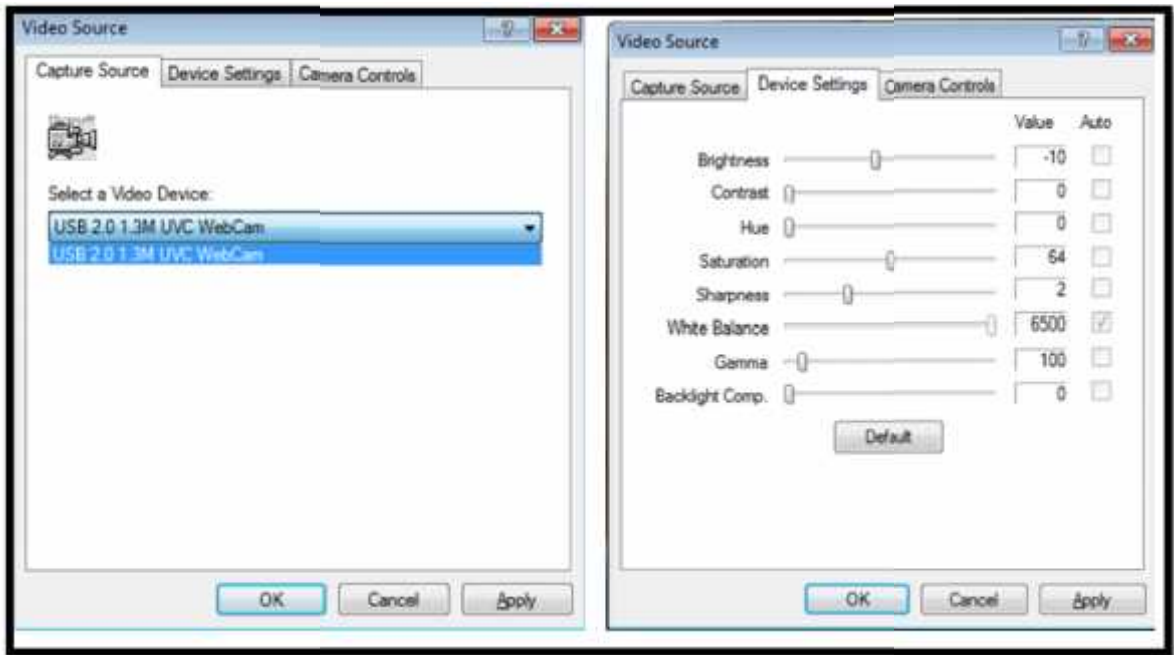


Figure 5.8 Video source dialogs

As shown in previous figure the Video Source dialog box controls the selection of video input channels and parameters affecting the video image being digitized in the frame buffer.

This dialog box enumerates the types of signals that connect the video source to the capture card (typically SVHS and composite inputs), and provides controls to change hue, contrast, or saturation.

Interfacing between PC and peer mobile

After save images and video from cameras, we need to send text and image to peer mobile in order to explain event occurred.

Initially, we must be established communication channel between PC and Peer mobile, the channel was created must be depend on MAC address of peer mobile, then initiate a new session to load image from specific location on PC and send it to peer mobile.


```

Private Sub Connect()

Dim add As String = "002345462Ef0" // MAC Address
Dim blue_add As BluetoothAddress =
BluetoothAddress.Parse(add)
Dim remoteEndPoint As BluetoothEndPoint = New
BluetoothEndPoint(blue_add, BluetoothService.ObexFileTransfer)
client = New BluetoothClient()
client.Connect(remoteEndPoint)
session = New ObexClientSession(client.GetStream(), UInt16.MaxValue)
session.Connect(ObexConstant.Target.FolderBrowsing)

End Sub

```

Here we show flow chart for monitoring unit

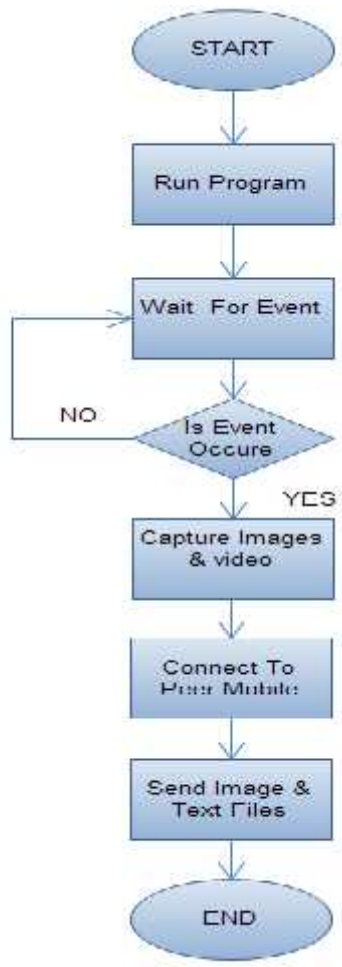


Figure 5.9 Monitoring Flow Chart

5.4 Mobile Programming

Firstly we need a mobile, which supports java requirements and MMS technology, this mobile must add the PC as Bluetooth paired device “receive from PC without user interaction”. Secondly you need to install NetBeans IDE (we used version

6.7.1) then create new project ,from file => new project => choose Java ME from categories => choose name and location of your project => select the device configuration (CLCD 1.1) and device profile (2.0) this depends on the mobile which will run the application on it .

First we import requires packages, next step is to make MIDlet class named SendMMS which extends MIDlet, after that we make another class named MMSFormSend.

First Class

Named SendMMS class extends MIDlet, consist of the following methods:

- Public void startApp() :
Signals the MIDlet that it has entered the *Active* state.
- Public void pauseApp() :
Signals the MIDlet to enter the *Paused* state
- public void destroyApp() :
Signals the MIDlet to terminate and enter the *Destroyed* state.
- public void exitMIDlet():
Signals the MIDlet to call destroyApp ().

Second Class:

The second class was an ordinary Java class named MMSFormSend extends Form and implements command listener. Now will explain its function:

In order to make system sense, mobile must checks its memory card continuously for the image, if not exist that's mean that the system is in safe mode and nothing happen, else image must exist in its memory card indicating events nature , mobile application and after founding image will read it also go to read text file and build the MMS to send it immediately. All that will implements in this class, first we

create a timer to be sure that the application will search for the image file continuously in predefined period suitable to our system consideration.

- initiate a timer equal to five seconds

```
int delay = 5000;  
int period = 5000;  
Timer timer = new Timer ();
```

- Then we will check the Memory Card for image file every five seconds

```
fCon =(FileConnection) Connector.open(file:///C:/predefgallery/predeffilereceived/mm.jpg");
```

- If the connection failed then do nothing, else sendMMS() method will be called

```
if (fCon.exists ()) {  
    SendMMS ();  
}
```

Inside sendMMS () method the following steps will occur:

- set up an MMS receiver address

```
String address = mms://+972599000000: "+appID;
```

- set up an mms connection

```
MessageConnection mmsconn = null;  
Mmsconn = (MessageConnection) Connector.open (address);
```

- Build the multipart message

```
MultipartMessage mmmessage = (MultipartMessage) mmsconn.newMessage  
(MessageConnection.MULTIPART_MESSAGE);
```

- set the address of the message

```
mmmessage.setAddress (address);
```

- set the subject of the message

```
mmmessage.setSubject("Your Emergency MMS ");
```

- Send MMS Message

```
mmsconn.send (mmmessage);
```

Some of libraries used:

- Set permissions for open connection

```
javax.microedition.Connector.*;
```

- Set permissions for send and receive operations

```
Javax.wireless.messaging.*;
```

Here we show flow chart diagram for mobile unit:

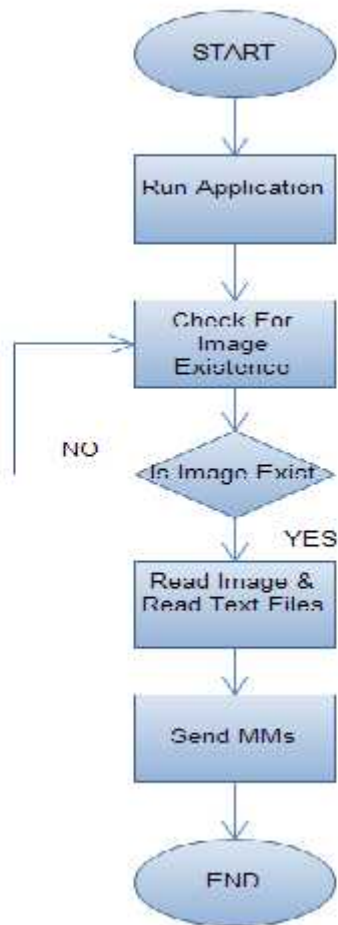


Figure 5.10 Mobile Flow Chart

6.1 Introduction

The whole testing stage will be described in this chapter, that's mean how to operate each part independently, in addition how to interface and test these parts together in order to achieve our goals.

6.2. Sensing unit

When we test sensing unit first we test every sensor separately from other sensor , and connected to circuit two LEDS (Red ,Yellow) to give indicate about the status in simply form (each led indicate one type of event) .As show in following figure.

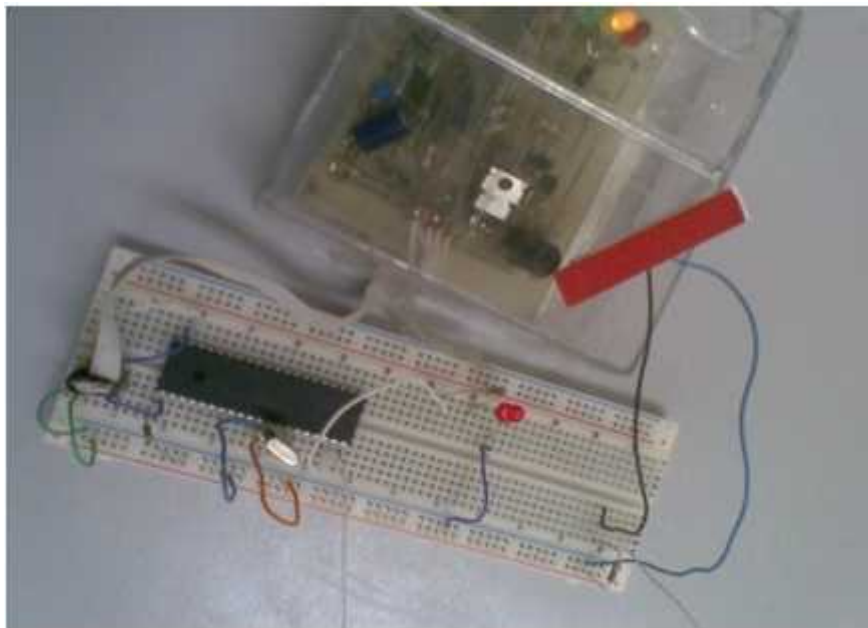


Figure 6.1 Interface Magnet To PIC

In our project we have two type of sensor ,analog and digital . In Interface analog sensor take care about configuration by open ADC ,then start conversion , after complete it should read value and update it to give temperature in Celsius degree.

But in digital sensor it give only ON and OFF status , so system wait until one or more of sensor go to ON status , then send appropriate signal to PC by serial cable. as shown in following figure.

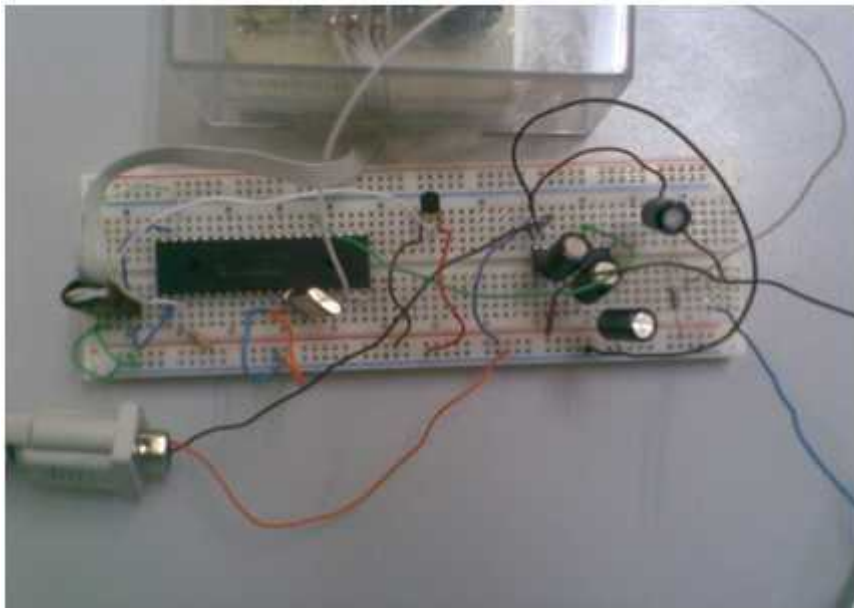


Figure 6.2 Interface Sensor To MAX232

In PC we receive data from PIC in program written by VB.net language ,before that we test serial port by Microsoft hyper terminal ,respond to us by out the value of event occur . we should configure Hyper terminal as in following
Start by type communicate name



Figure 6.3 Connection Name

Previous figure show name of connection .Then configure port as shown In the following Figure.



Figure 6.4 Configuration of port

Example of output status as show in following figure :

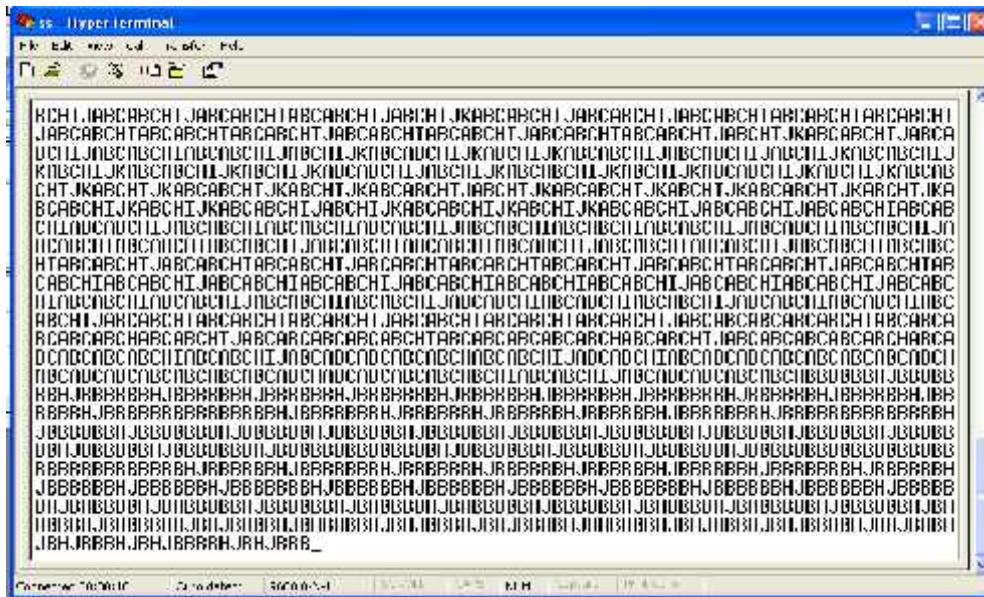


Figure 6.5 Status on terminal .

In vb.net code ,first do interface to read from serial port as in following :

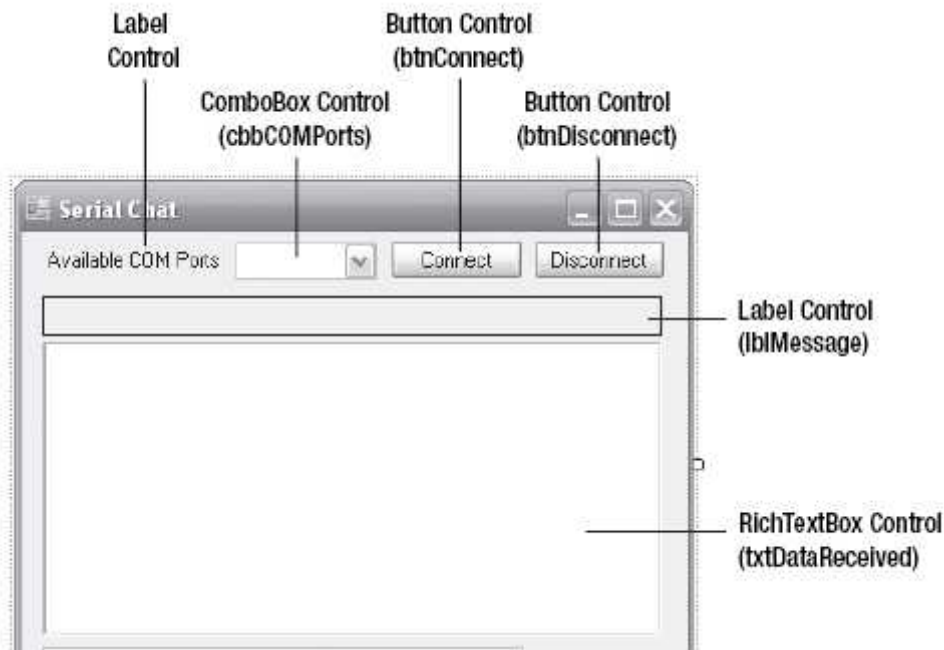


Figure 6.6 Interfaces To Serial Port

Data Received from serial port and appear it in RichTextBox as show in figure 6.6 .after that make program to test system status depend on command word .

6.3 Monitoring unit

After we connected cameras figure 6.7 to our system and run program on visual basic .NET , shows us video source dialog as shown in figure 6.8 to add video devices to our program , then as shown in figure 6.9, the stream of video comes from cameras is showing in video windows.

Then the program must wait until receive event signal from serial port to take images and video that store locally in PC.



Figure 6.7 camera device

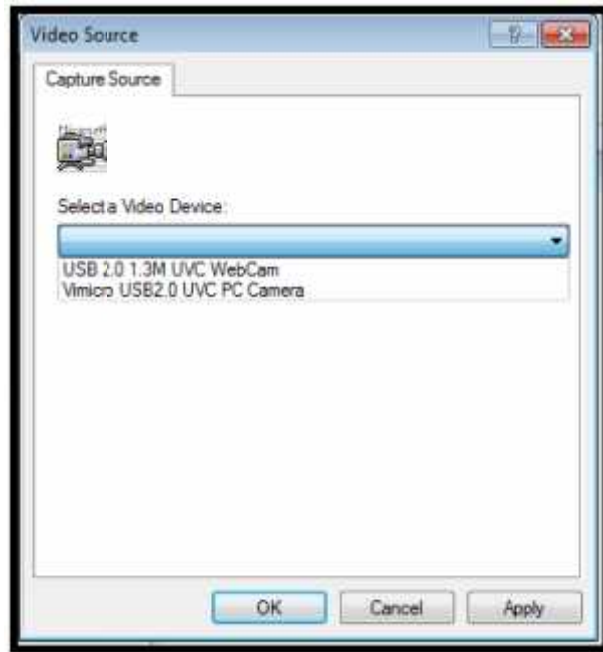


Figure 6.8 video source dialog

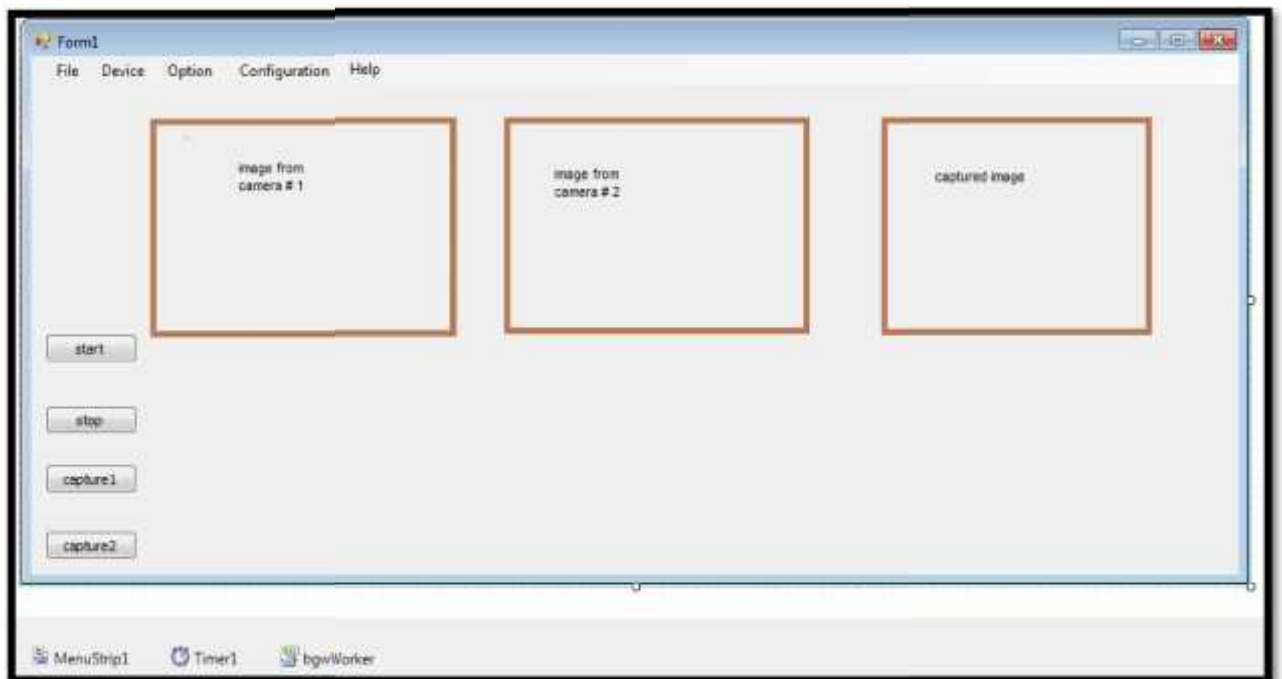


Figure 6.9 camera form program

Problems

If we have one video device in our system we don't need user interaction to add cameras to our program which mean there is no appearance for video source dialog , but if we have more than one device (as we need in our project) we notice the appearance of the dialog.

The driver of video device is showing this dialog and we cannot find a method to control its appearance, so in our project the user must add cameras initially to program then, our program wait for event signal.

PC to peer mobile

How send image and text from PC to peer mobile?

Initially we must add PC as Bluetooth paired device on peer mobile as shown in figure 6.10 then our program send image and text to peer mobile using its MAC address as shown on the following figure.

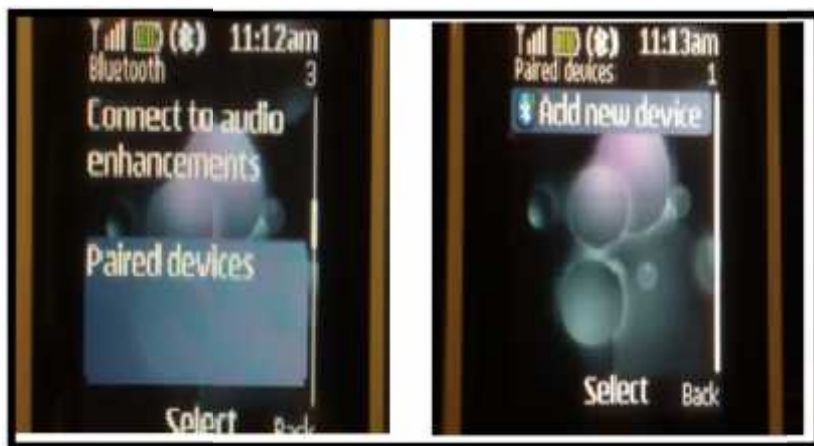


Figure 6.10 Mobile paired device

Challenges:

Initially we connect peer mobile by USB cable to PC in order to make a communication between them but we cannot find a method that enable us to send image and text from PC and receive them from mobile, so we moving to another way in order to achieve our goal, we find OTA (Of The Air message) that allow us to send picture message from pc to mobile but we find that message cannot carry more than 256 byte picture message.

After that we go to Bluetooth solution and we solve our problem

6.4 Mobile unit

To implement and test implementation in previous chapter, we put the SIM card and connect the peer mobile to PC through Bluetooth technology, then start to test step by step:

First step we done to test our application was try to run simple applications in order to test send an SMS message only.



Figure 6.11 SMS On Simulator

Second was try to read an image and display it on simulator screen in order to be sure that our application can deal with the image comfortably.

At the first time, there was an exception appeared in simulator screen, that's mean that we can't read the image .Finally we place the image want to read in the same application package " in resource file " which mean that this image packaged inside JAR file, by doing that we can read and display the image on the simulator screen.



Figure 6.12 Image On Simulator

After that we go to test another component of the application ,it was Timer. We try to choose the most suitable timer value in order to get real time or near real time response, we find that 5 seconds was fine.

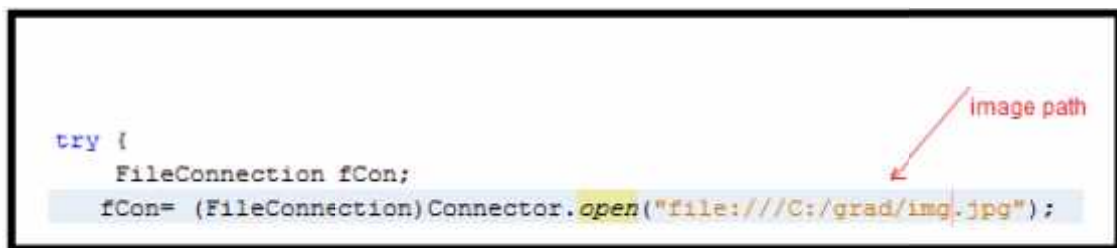
```
int delay= 5000;  
int period= 5000;  
Timer timer = new Timer();  
timer.scheduleAtFixedRate(new TimerTask(){
```

The next step was actually very hard one , its take a lot of time. We now move to read the image from another platform on the mobile, the application try to catch the image and the text files and read both from the mobile memory.

First goals of j2ME application was to check for the existence of these files , then if they exist read them.

Here we faced a problem of how we can reach and read saved image, we used FileConnection in order to attach this image and check for its existence ,then read it.

```
try {
    FileConnection fCon;
    fCon= (FileConnection)Connector.open("file:///C:/grad/img.jpg");
```



Then create this folder " grad " and copy the image " img " inside. When the program run they print on the simulator screen " NotFound " string indicates that J2ME application can't find target file as shown in the next figure, although we make sure that it is exist in the correct directory.



Figure 6.13 Mobile Testing Result

Here we try tens of methods to solve this problem which is J2ME programming related, but without any result. Also look for any suggestion on internet, actually there were more than one solution, once said that we must use a form for the path like this "file:/// ", so this form was used but no result appeared all known ways tried, but without any effectiveness unless the solution had appeared, the problem was that the J2ME application can't see all parts of the computer, it can see just a folder named "root folder" this folder must be created manually in long folder series before run the application, otherwise the application will not see target file. This folder is created by following next steps:

Go to javame-sdk which installed in the beginning of using J2ME "java SDK". There, create new folder named "3.0". Inside 3.0 folder create another new folder named "0". Another new folder will be created inside 0 folder, named "appdp". Here create another folder named "filesystem". Inside the filesystem create the last folder named "root1".

Inside this folder "root1" put any file want your J2ME application to see and read it, so we conclude that the J2ME application deal with constant unique path called its application root, inside this root everything is accessible, any path else will be not useful.

The next step was, how to find the root of the mobile, we know now the application root on PC in order to test application, also it is most necessary to find out application root on the mobile needed to run the application on.

Here we found nothing about mobile application root, also application was print out "NOT founded" on the mobile screen. as shown on the next figure.



Figure 6.14 Mobile Root Problem

so we enforced to use our methods to check out the mobile application root. Another J2ME application was created , but this time it has different job. The following is a sub code of the second J2ME application.

```
Enumeration drives = FileSystemRegistry.listRoots();
System.out.println("The valid roots found are: ");
while(drives.hasMoreElements()) {
    String root = (String) drives.nextElement();
    form.append(root);
    // append(root);
    System.out.println("\t"+root);
}
```

This new J2ME application have the function of mobile root finder. First it runs on PC in order to check if it work properly, it print out " The Valid Root Found are : root1/ " on the simulator screen. See figure 6.15.



Figure 6.15 PC Root

Next it run on the mobile and print out its roots, yes the mobile have more than one root , J2ME application deal with mobile memories just like root1 in PC , so it deal with the mobile memory as first root named "C : " , and with the memory Card as the second root named " E : " , as shown in Figure 6.16. by this way we can solves our roots big problem, so we now stand on the first element of the image/text files path.



Figure 6.16 Mobile Roots

Now we have the image, the next step is how to send it as MMS message : we set up the address and the subject of MMS to send our message.

```
String appID = midlet.getAppProperty("MMS-ApplicationID");
String address = "mms://+972599041720:" + appID;
mmessage.setSubject("Your Emergency MMS");
mmsconn.send(mmessage);
```

And now our application is ready to test in its terminal shape on mobile. We install the application and start it from the game menu where it is installed.



Figure 6.17 Application On Mobile Screen

In this stage we faced another problem , this was an authentication problem. When application run on the mobile, an authentication string is print out on the mobile screen, it indicates user to allow application to :

- Read user data
- Write and edit user data
- Send MMS message

Also its indicates user that this application is not from trusted supplier. This is shown in the next figures.

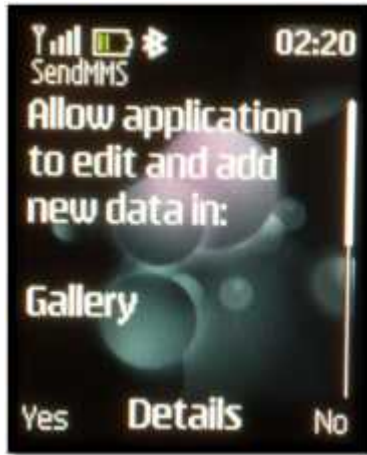


Figure 6.18 Write Authentication Problem

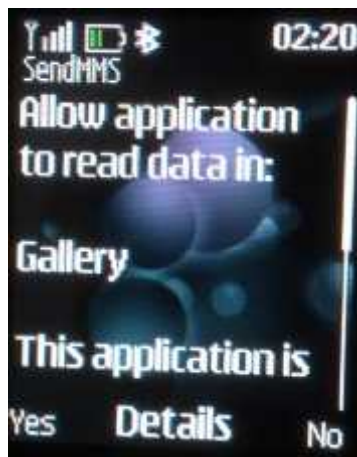


Figure 6.19 Read Authentication Problem

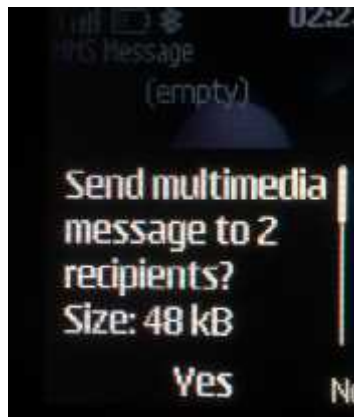


Figure 6.20 Send MMS Authentication Problem



Figure 6.21 Not Trusted Application

The authentication problem was a big seriously problem, we tried hard to solve it . first we look for a solution in the same mobile , look for application details and properties, found something called application access " app.access " see Figure 6.22.

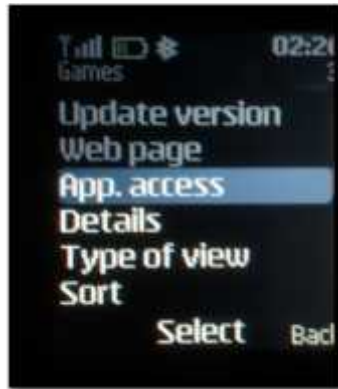


Figure 6.22 Phone Application Access

In this list, three options were found:

communication,

Data access,

And Auto start, see Figure 6.23



Figure 6.23 Application Data Access

Then, in Data access option something interesting was found. There were an access control on each of operation, we interesting in two of them. Look at Figure 6.24.



Figure 6.24 : Data Access Options

First one was Read User Data option, inside this option another sub list contained four option:

- Ask Every Time
- Ask First Time Only
- Always Allowed
- Not Allowed , see figure 6.25.

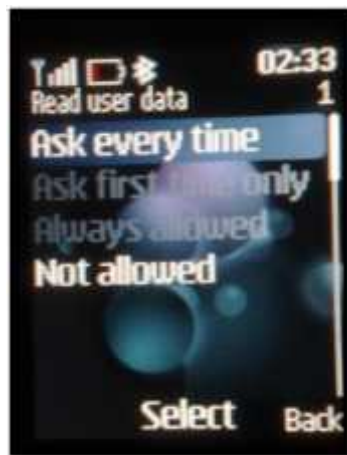


Figure 6.25 Data Access Option 1

Second, was Add And Edit user Data option, inside this option another sub list contained also four option:

- Ask Every Time
- Ask First Time Only
- Always Allowed
- Not Allowed , see figure 6.26



Figure 6.26 Data Access Option 2

Unfortunately, second and third option is not available for us, as shown in the last figure. So by using mobile sitting ,the problem can't solved yet.

Sign MIDlet:

We'll show how to sign a MIDlet Suite to grant the execution of operations that usually the midlet may only run after a user confirmation.

WTK: The Sun J2ME Wireless Toolkit supports the development of Java applications that run on devices such as cellular phones .We'll use WTK to sign our application.

Main Concepts:

Permission: is an operation or method that the application can execute inside its runtime environment.

Protection domain: is a set of permissions.

Keystore: which is a repository of certificates . As we'll see later, when we install a signed midlet into a device, it checks for a certificate that can be used to recognize the midlet suite as signed. If no certificate is found, the midlet isn't secure and the device can advise the user to not install it. On the contrary, if one suitable certificate is found, the device keystore can be used for verifying that the midlet suite is signed, and thus we have what we call a trusted midlet suite.

Trusted and Untrusted Midlet Suites:

The trusted ones are linked to a certificate and so they have all the permissions included in the associated protection domain. The untrusted ones are linked to a domain which is often called untrusted domain which has no specific permissions.

How can a device know if a midlet suite is trusted or untrusted? Well, if the midlet suite is signed with a specific certificate, the device will search, into its keystore, for a suitable certificate. The subsystem in charge of associating a specific certificate to a particular protection domain is provided by the device vendor, and so it's a vendor-specific process. On general terms, we can say that every protection domain is linked to a root certificate the device itself uses to check signed applications. When an application developer signs a midlet suite, he or she can use different certificates released by different certificate authorities. The protection

domain linked to the midlet suite will be the one the first matching root certificate recognizes. This means that a midlet suite can be linked to only one protection domain.

How to set the permissions for a midlet suite :

From the user interaction point of view, MIDP 2.0 specification describes two different kinds of permissions:

- Allowed permission
- User permission

Allowed permissions are those a trusted midlet suite can execute without an explicit user interaction. This kind of permissions are assigned to a signed midlet suite by its protection domain. User permissions requires an interaction with the user. So, if we want a midlet suite to execute a protected API without any user interaction, we have to sign our application using a certificate that the device links to a protection domain in which the API is marked as allowed.

On the other hand, if the midlet suite is linked to a protection domain in which the protected API permission is marked as "user permission", the application will ask the user if he or she wants to execute it or not.

Following steps show how sign our midlet using WTK

- 1- Loading JAD file to WTK.
- 2- Choose trusted domain using edit/preference/security (Fig6_27).

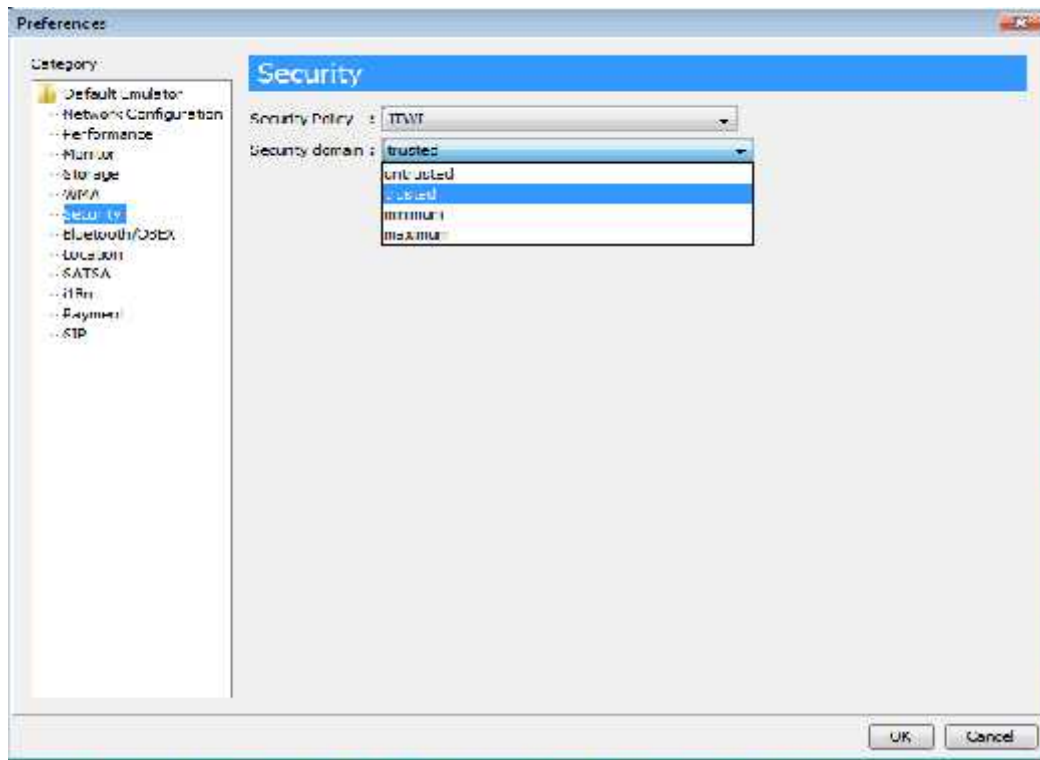


Figure 6.27 Trusted Protection Domain

- 3- Generate a Key pair to sign the midlet suite using Keystore/New Key Pair option of our tool (Fig6_28).



fig. 6_28 generate New Key

Here we can insert information about the company and the alias name. When we press ok we have the important step. Here the tool shows us the list of available protection domains we can associate to the key we're generating. It's a fundamental step, since the protection domain we choose here will be the one associated to the midlet when it'll be executed . We choose the trusted one as shown on the following figure.

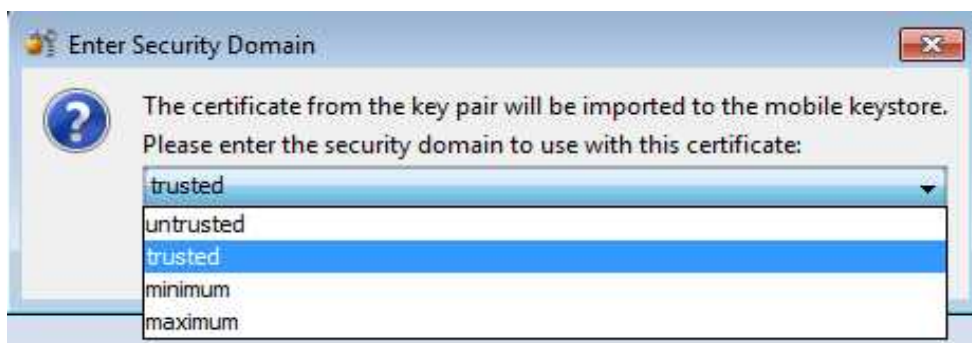


Figure 6.29 Trusted Certificate Domain

- 4- All we have to do now is to sign the midlet suite with the Action/Sign Midlet Suite option of the tool. Here we want to sign and then confirm. The midlet is now signed and associated to the trusted domain.

5- Run Midlet and install it to peer mobile , after that we show that all option became active

When we execute now our midlet into the peer mobile - the midlet suite being associated to the trusted domain - the MMS operation can be done without user confirmation.

7.1 Introduction

The project that has been done was a step for developing the idea of remote monitor home. Also the project was a good step in developing smart houses. Meanwhile we have some recommendations and suggestions for the future work. The following section will discuss them.

7.2. System Achievements

Almost all the goals of our system have been achieved. In this point the main achievements of the system are discussed and the ways of achieving it.

We build sensing unit that cover many event natures (motion, smoke, door /windows, and temperature) in order to discover any abnormal action happen inside home.

We can connect more than video device to our system in order to capture images and record video from different corner and we can send image and text using Bluetooth to mobile unit.

We can run our application continuously on peer mobile in order to achieve project goals to get real time or near real time response as soon as the image is arrived on to mobile memory by bluetooth, this application will catch it and send it immediately to the owner, which mean our project is success to alert home owner.

7.3. Real Learning Outcomes

After the implementation of the project we have an expert in the following points:

- Learn how to use and program 18F4550 microcontroller.
- How to remotely monitor device.
- learn how to program video devices using visual basic.NET

- Bluetooth programming.
- Mobile programming.
- Faces many problems with communicating with the mobile and learn how to solve it.

7.4. Recommendations

After our work on this project and after facing many problems during the implementation, we as a project team, see that the following points may be a good improvement for this project in order to make it more sense and more reliable:

- Increase number of cameras to avoid system failure when one camera is damaged.
- Increase number of sensing unit.
- Add another sensor type.
- Increase number of MMS receiver.
- Send video record to owner.
- Send voice record.

```
Imports System.Drawing.Imaging
Imports System.Runtime.InteropServices, System.IO
```

```
Imports System.Data
Imports WebCam_Capture
Imports System
Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.Drawing
Imports System.Net.Sockets
Imports System.Windows.Forms
```

```
Imports InTheHand.Net
Imports InTheHand.Net.Bluetooth
Imports InTheHand.Net.Sockets
Imports InTheHand.Windows.Forms
Imports Brecham.Obex
Imports Brecham.Obex.Objects
```

```
Imports System.IO.Ports.SerialPort
Imports System.IO.Ports
```

```
Public Class Form1
```

```
    Private WithEvents serialPort As New IO.Ports.SerialPort
    Dim by As Byte
```

```
    Dim file, filename As String
    Dim w As Integer = 20
    Dim i As Integer
```

```
    Public Structure CAPTUREPARMS
        Dim dwRequestMicroSecPerFrame As Integer
        Dim fMakeUserHitOKToCapture As Boolean
        Dim wPercentDropForError As Integer
        Dim fYield As Boolean
        Dim dwIndexSize As Integer
        Dim wChunkGranularity As Integer
        Dim fUsingDOSMemory As Boolean
        Dim wNumVideoRequested As Integer
        Dim fCaptureAudio As Boolean
        Dim wNumAudioRequested As Integer
        Dim vKeyAbort As Integer
        Dim fAbortLeftMouse As Boolean
        Dim fAbortRightMouse As Boolean
        Dim fLimitEnabled As Boolean
        Dim wTimeLimit As Integer
        Dim fMCIControl As Boolean
        Dim fStepMCIDevice As Boolean
        Dim dwMCIStartTime As Integer
        Dim dwMCIStopTime As Integer
        Dim fStepCaptureAt2x As Boolean
        Dim wStepCaptureAverageFrames As Integer
        Dim dwAudioBufferSize As Integer
        Dim fDisableWriteCache As Boolean
        Dim AVStreamMaster As Integer
    End Structure
End Class
```



```

End Structure

Declare Function SetWindowPos Lib "user32" Alias "SetWindowPos" _
    (ByVal hwnd As Integer, ByVal hWndInsertAfter As Integer, ByVal x
As Integer, _
    ByVal y As Integer, ByVal cx As Integer, ByVal cy As Integer,
ByVal wFlags As Integer) As Integer

    '--This function destroys the specified window--
Declare Function DestroyWindow Lib "user32" (ByVal hwnd As Integer)
As Boolean

    '--This function sends the specified message to a window or
windows--

Declare Function SendMessage Lib "user32" Alias "SendMessageA"
    (ByVal hwnd As Integer, ByVal wMsg As Integer, ByVal wParam As Short,
ByVal lParam As String) As Integer

Declare Function SendMessage2 Lib "user32" Alias "SendMessageA"
    (ByVal hwnd As Integer, ByVal wMsg As Integer, ByVal wParam As Short,
ByRef lParam As CAPTUREPARMS) As Integer

Private webcam As WebCam
Dim caprams As New CAPTUREPARMS
Dim hwnd As Integer

Const WM_CAP_START = &H400S
Const WS_CHILD = &H40000000
Const WS_VISIBLE = &H10000000

Const WM_CAP_DRIVER_CONNECT = WM_CAP_START + 10
Const WM_CAP_DRIVER_DISCONNECT = WM_CAP_START + 11
Const WM_CAP_SEQUENCE = WM_CAP_START + 62
Const WM_CAP_SET_SEQUENCE_SETUP = WM_CAP_START + 64
Const WM_CAP_SET_SCALE = WM_CAP_START + 53
Const WM_CAP_SET_PREVIEWRATE = WM_CAP_START + 52
Const WM_CAP_SET_PREVIEW = WM_CAP_START + 50
Const WM_CAP_STOP = WM_CAP_START + 68
Const WM_CAP_FILE_SET_CAPTURE_FILE = WM_CAP_START + 20
Const WM_CAP_ABORT = WM_CAP_START + 69

Const SWP_NOMOVE = &H2S
Const SWP_NOSIZE = 1
Const SWP_NOZORDER = &H4S
Const HWND_BOTTOM = 1
Private mintCount As Integer = 3
Dim client As BluetoothClient
Dim previousItemsStack As Stack(Of ListViewItem()) = New Stack(Of
ListViewItem())
Dim session As ObexClientSession

Private Sub btnStart_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnStart.Click
    webcam.Start()

```

```
End Sub
```

```
Private Sub bntStop_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles bntStop.Click  
    webcam.Stop()  
End Sub
```

```
Private Sub bntCapture_Click(ByVal sender As System.Object, ByVal e  
As System.EventArgs) Handles bntCapture.Click
```

```
    imgCapture.Image = imgVideo1.Image
```

```
End Sub
```

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles MyBase.Load
```

```
    With caprams
```

```
        .AVStreamMaster = 1 'maybe 1?  
        .dwAudioBufferSize = 10  
        .dwIndexSize = 34952  
        .dwMCIStartTime = 0  
        .dwMCIStopTime = 0  
        .dwRequestMicroSecPerFrame = 16667  
        .fAbortLeftMouse = False  
        .fAbortRightMouse = False  
        .fCaptureAudio = True  
        .fDisableWriteCache = False  
        .fLimitEnabled = True  
        .fMakeUserHitOKToCapture = False  
        .fMCIControl = False  
        .fStepCaptureAt2x = False  
        .fStepMCIDevice = False  
        .fUsingDOSMemory = False  
        .fYield = True  
        .vKeyAbort = Nothing  
        .wChunkGranularity = 0  
        .wNumAudioRequested = 10  
        .wNumVideoRequested = 32  
        .wPercentDropForError = 0  
        .wStepCaptureAverageFrames = 0  
        .wTimeLimit = 300 '# of seconds of video to caputre!!
```

```
    End With
```

```
    webcam = New WebCam()  
    webcam.InitializeWebCam(imgVideo1)
```

```
    webcam.Start()  
    PreviewVideo(imgVideo2)  
    capture_img()
```

```
End Sub
```

```

Private Sub PreviewVideo(ByVal pbCtrl As PictureBox)

hWnd = WebCam_Capture.WebCamCapture.capCreateCaptureWindowA(0,
WS_VISIBLE Or WS_CHILD, 0, 0, 0, 0, imgVideo2.Handle.ToInt32, 0)

If WebCam_Capture.WebCamCapture.SendMessage(hWnd,
WM_CAP_DRIVER_CONNECT, 0, 0) Then

'---set the preview scale---
WebCam_Capture.WebCamCapture.SendMessage(hWnd, WM_CAP_SET_SCALE, True,
0)
'---set the preview rate (ms)---
WebCam_Capture.WebCamCapture.SendMessage(hWnd, WM_CAP_SET_PREVIEWRATE,
30, 0)
'---start previewing the image---
WebCam_Capture.WebCamCapture.SendMessage(hWnd, WM_CAP_SET_PREVIEW,
True, 0)
'---resize window to fit in PictureBox control---
SetWindowPos(hWnd, HWND_BOTTOM, 0, 0, pbCtrl.Width, pbCtrl.Height,
SWP_NOMOVE Or SWP_NOZORDER)

Else
'---error connecting to video source---
DestroyWindow(hWnd)
End If
End Sub

Private Sub capture_img()

    Dim i As Integer
    Dim e As EventArgs

    For i = 0 To 4 Step 1

        System.Threading.Thread.Sleep(1000)
        bntCapture_Click(Capture, e)
        System.Threading.Thread.Sleep(1000)

        file = DateTime.Now.ToString("yyyyMMdd") & "_" &
DateTime.Now.ToString("HHmmss")
        filename = "d:\\monitor\" & file & ".bmp"
        imgCapture.Image.Save(filename, ImageFormat.Bmp)

    Next

    filename = "d:\\monitor\1.jpeg"
    imgCapture.Image.Save(filename, ImageFormat.Jpeg)

    send_image()
    webcam.Stop()
    record_video()

End Sub

```

```

Private Sub record_video()

    file = DateTime.Now.ToString("yyyyMMdd") & "_" &
DateTime.Now.ToString("HHmmss")
    filename = "d:\\monitor\" & file & ".avi"

    SendMessage2(hWnd, WM_CAP_SET_SEQUENCE_SETUP, Len(caprams), caprams)
    SendMessage(hWnd, WM_CAP_FILE_SET_CAPTURE_FILE, 0, filename)
    SendMessage(hWnd, WM_CAP_SEQUENCE, 0, CType(0, String))

    System.Threading.Thread.Sleep(300000)

End Sub

Private Sub send_image()

    Connect()
    System.Threading.Thread.Sleep(1000)

    UploadFiles(" d:\\monitor\1.jpeg")
    System.Threading.Thread.Sleep(3000)

End Sub

Private Sub VideoFormat_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles VideoFormat.Click
    webcam.ResolutionSetting()
End Sub
.....
Private Sub VideoSource_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles VideoSource.Click
    webcam.AdvanceSetting()
End Sub
.....
Private Sub ExitToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ExitToolStripMenuItem.Click
    Me.Close()
End Sub
.....
Private Sub AboutToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
AboutToolStripMenuItem.Click

End Sub
.....
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick

    Form1_Load(sender, e)
    ' serial_config()
    Timer1.Enabled = False

End Sub

```

```

.....
Private Sub Connect()

    Dim add As String = "002345462Ef0"
    Dim blue_add As BluetoothAddress = BluetoothAddress.Parse(add)
    Dim remoteEndPoint As BluetoothEndPoint = New
BluetoothEndPoint(blue_add, BluetoothService.ObexFileTransfer)
    client = New BluetoothClient()
    client.Connect(remoteEndPoint)
    session = New ObexClientSession(client.GetStream(),
UInt16.MaxValue)
    session.Connect(ObexConstant.Target.FolderBrowsing)

End Sub
.....

Private Sub UploadFiles(ByVal files As String)

    Dim size As Long = 0
    Dim filestoupload As List(Of String) = New List(Of String)()
    Dim info As FileInfo = New FileInfo(files)
    filestoupload.Add(files)
    size += info.Length

    Dim download As Boolean = False
    Dim filesProcessed As Integer = 0
    Dim progress As Long = 0
    Dim i As Integer
    Dim start As DateTime = DateTime.Now

    Dim currentfile As String = filestoupload(0)
    bgwWorker.ReportProgress(Int(((progress * 100) / size)), i + 1)

    Dim filename As String
    If (download) Then
        filename = Path.Combine(Dir, currentfile)
    Else
        filename = currentfile
    End If

    Dim hoststream As FileStream
    hoststream = New FileStream(filename, FileMode.Open,
FileAccess.Read, FileShare.None)

    Dim remotestream As AbortableStream = Nothing

    remotestream = session.Put(Path.GetFileName(currentfile), Nothing)

    Using (hoststream)

        Using (remotestream)

            Dim result As Long
            result = ProcessStreams(hoststream, remotestream, progress,
currentfile)

```

```
        If (result <> 0) Then
            progress = result
        End If
    End Using
End Using

End Sub
```

```
.....

Private Sub Disconnect()
```

```
    session.Disconnect()
    session.Dispose()
    session = Nothing
    client.Close()
    client.Dispose()
    client = Nothing
```

```
End Sub
```

```
.....

Private Function ProcessStreams(ByVal source As Stream, ByVal
destination As Stream, ByRef progress As Long, ByVal filename As
String) As Long
```

```
    Dim buffer(1024 * 4) As Byte
    While (True)
        Dim length As Integer = source.Read(buffer, 0, buffer.Length)
        If (length <> 0) Then
            destination.Write(buffer, 0, length)
            progress += length
        Else
            Exit Function
        End If

    End While
    Return progress
```

```
End Function
```

```
.....

Private Sub serial_config()
```

```
    If serialPort.IsOpen Then
        serialPort.Close()
    End If
    Try
        '---configure the serial port with the various
        ' parameters---
        With serialPort
```

```
        .PortName = "COM20"  
        .BaudRate = 9600  
        .Parity = IO.Ports.Parity.None  
        .DataBits = 8  
        .StopBits = IO.Ports.StopBits.One  
        .Handshake = IO.Ports.Handshake.None  
    End With  
    '---open the serial port---  
    serialPort.Open()  
Catch ex As Exception  
    MsgBox(ex.ToString)  
End Try
```

End Sub

.....

```
Private Sub serialPort_DataReceived(ByVal sender As Object, ByVal e  
As System.IO.Ports.SerialDataReceivedEventArgs) Handles  
serialPort.DataReceived  
    by = serialPort.ReadByte  
    serialPort.DiscardInBuffer()  
  
    TextBox1.BeginInvoke(New myDelegate(AddressOf updateControl),  
New Object() {})  
    serialPort.Close()  
    webcam.Stop()
```

End Sub

.....

```
Public Delegate Sub myDelegate()  
  
Public Sub updateControl()  
  
    Try  
        If by = 65 Then  
            TextBox1.Text = "Motion detector "  
        End If  
  
        If by = 66 Then  
            TextBox1.Text = "Door / window open "  
        End If  
  
    Catch ex As Exception  
        MsgBox(ex.ToString)  
    End Try
```

End Sub

End Class

////////////////////////////////////

```

#include<p18f4550.h>
#include<delays.h>
#include<adc.h>
#include<usart.h>

#define mot PORTDbits.RD0
#define mag PORTDbits.RD1
#define smk PORTDbits.RD2

#pragma config WDT=OFF
#pragma config LVP=OFF
#pragma config FOSC=INTOSC_HS

void main()
{
int temp =0;
unsigned int t_val=0;

OSCCON=0b01111110;

OpenADC(ADC_FOSC_64 & ADC_RIGHT_JUST & ADC_2_TAD , ADC_CH0 &
ADC_INT_OFF & ADC_REF_VDD_VSS , ADC_2ANA);

OpenUSART( USART_TX_INT_OFF & USART_RX_INT_OFF & USART_ASYNCH_MODE
& USART_EIGHT_BIT & USART_CONT_RX & USART_BRGH_LOW, 12 );

PORTD=0;
PORTC=0;
PORTA=0;

TRISC=0;
TRISD=255;
TRISA= 1;

while(1)
{

ConvertADC();
while(BusyADC());
t_val= ReadADC();
Delay10TCYx(0);

temp = ((t_val * 500) /1023);

if (mot) // motion -----1
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('A');
}
}

```



```
else if (!smk) // smoke -----2
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('C');
}
```

```
else if (mot && !smk) // -----3
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('C');
}
```

```
else if ( temp > 35 ) // LM -----4
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('D');
}
```

```
else if (temp >35 && mot) // -----5
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('E');
}
```

```
else if (temp >35 && !smk)//-----6
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('F');
}
```

```
else if (temp>35 && mot && !smk)//-----7
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('G');
}
```

```
else if (mag ) // magnat -----8
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('B');
}
```

```
else if (mag && mot) // -----9
```

```

{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('I');
}

else if (mag && !smk) //-----10
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('J');
}

else if (mag && mot && !smk)//----11
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('K');
}

else if (mag && temp>35)//-----12
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('L');
}

else if (mag && temp >35 && mot) //-----13
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('M');
}

else if (mag && temp >35 && !smk) //-----14
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('N');
}

else if (mag && temp >35 && mot && !smk) // -----15
{
while(BusyUSART());
Delay10TCYx(0);
WriteUSART('O');
}}
CloseUSART();
}

```

References:

- [1] <http://www.tradercity.com/board/products-1/offers-to-sell-and-export-1/projects-on-home-security-using-mobile-phone-107195> [Accessed September 8 2010]
- [2] <http://www.plainfonet.com/vb/showthread.php?t=464> [Accessed September 30 2010]
- [3] Jon S. Wilson, Sensor Technology Handbook, Elsevier, 30 Corporate Drive, suite 400, Burlington, MA 01803, USA; 2005, Elsevier Inc.
- [4] http://hobby_elec.piclist.com/e_pic.htm [October 3 2010]
- [5] [http:// www.beyondlogic.org/serial/serial.htm#30](http://www.beyondlogic.org/serial/serial.htm#30) [October 3 2010]
- [6] <http://electronic.howstuffworks.com/cell-phone1.htm> [Accessed October 3 2010]
- [7] <http://en.wikipedia.org/wiki/GSM> [Accessed October 12 2010]
- [8] <http://www.wisegeek.com/what-is-sms.htm> [Accessed October 12 2010]
- [9] <http://en.wikipedia.org/wiki/SMS> [Accessed October 27 2010]
- [10] http://en.wikipedia.org/wiki/Multimedia_messaging_Service [Accessed November 13 2010]
- [11] <http://en.kioskea.net/contents/bluetooth/bluetooth-intro.php3> [Accessed November 15 2010]