East West University

Department of Electrical and Electronic Engineering

Project Report on

Microcontroller Based Smart Home Monitoring and Control System

By

G. M. Hasan-Ul-Banna 2009-2-80-022

Md. Shafayatul Haque 2009-2-80-029

Mohammad Towfiqul Hasan Chowdhury 2009-2-80-031

Submitted to the

Department of Electrical and Electronic Engineering

Faculty of Sciences and Engineering

East West University

Dhaka, Bangladesh

In partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Electronic Engineering (B.Sc. in EEE)

Spring 2013

Approved by

……....

Project Supervisor:

Dr. Muhammed Mazharul Islam
Assistant Professor
Electrical & Electronic Engineering
East West University

……..……..

Department Chairperson:

Mohammad Mojammel Al Hakim
Chairperson & Associate Professor
Electrical & Electronic Engineering
East West University

Approval

The project work titled ‘Microcontroller Based Smart Home Monitoring and Control System’ submitted by G. M. Hasan-Ul-Banna (2009-2-80-022), Md. Shafayatul Haque (2009-2-80-029) and Md. Towfiqul Hasan Chowdhury (2009-2-80-031) in Spring, 2013 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelor of Science in Electrical and Electronic Engineering.

……

Department Chairperson:

Mohammad Mojammel Al Hakim
Chairperson & Associate Professor
Electrical & Electronic Engineering
East West University

Acknowledgment

First of all, we are grateful to the almighty Allah for giving us this opportunity to complete the project.

We would like to thank Dr. Muhammed Mazharul Islam, Assistant Professor, Department of Electrical and Electronic Engineering (EEE), East West University (EWU), Dhaka, our supervisor, for his constant guidance, supervision, constructive suggestions and constant support during this project work.

We are grateful to Dr. Mohammad Mojammel Al-Hakim, Associate Professor and Chairperson, Dr. Anisul Haque, Professor, and Dr. Halima Begum, Assistant Professor, Department of Electrical and Electronic Engineering (EEE), East West University (EWU), for their valuable suggestions and help. We would also like to thank our academic advisors Mr. Md. Niazul Islam Khan, Mr. Rizvi Ahmed and also other faculty members, office staffs and EWU in general.

At last we want to thank our parents and all our friends for their moral support and helpful discussion during this project work.

Authorization

We hereby declare that we are the sole authors of this project. We authorize East West University to lend this project to other institutions or individuals for the purpose of scholarly research.

…….. ..…..……

Project Supervisor:
Dr. Muhammed Mazharul Islam

G. M. Hasan-Ul-Banna

...…….

Md. Shafayatul Haque

….……...……

Mohammad Towfiqul Hasan Chowdhury

We further authorize East West University to reproduce this project report by photocopy or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

…….. ..…..……

Project Supervisor:
Dr. Muhammed Mazharul Islam

G. M. Hasan-Ul-Banna

...…….

Md. Shafayatul Haque

….……...……

Mohammad Towfiqul Hasan Chowdhury

East West University

Dhaka, Bangladesh

April, 2013

Abstract

In our home we need security guards for security purpose, who are not always reliable. Fire hazard is another security concern of house, and damage may occur due to accidental fire.

For a physically handicapped person, access to switches for controlling electrical fixture such as lights/fans may be difficult.

In our house, parking lots are personally allocated for individual users so parking management is very easy. Situation is not the same for a market or a big institution. It creates traffic jam at entrance and is a time consuming concern.

To solve these problems we designed a system where security system of main entrance is password protected. A fire alarm system is installed to detect fire hazard. We have remote control access to control fixtures which helps physically handicapped users. To solve parking problem we designed a mapping board which is installed just in front of parking slots. It displays the slots condition of the garage (i.e. empty or occupied) and continuously updates its slot information.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1 Objective

In our home, we have to depend on the guards for the security of the house. Sometimes these persons are not reliable, and employing them may also be expensive. An effective solution to this is a security system that does not rely on another person's good character or willingness. Even if a guard is employed, a secure electronic lock provides an extra level of protection.

Another safety concern in any house is fire hazard. If a fire occurs in the house, it might not be possible to detect it until much damage has been done. So, a fire alarm incorporated in the security system of the house greatly reduces the chance of damages to people and property.

Inside our home, whenever we want to turn a light or fan on/off, we walk up to the switchboard and push a switch. This is not so difficult for a physically fit person being able to walk. For a physically handicapped person, however, this simple task of pushing a switch to turn on/off a light or fan at their own will may be difficult to perform, and s/he may have to be dependent on someone else for doing this menial job. The same thing may happen when they want to open or close a window. If these tasks could be performed using a remote control, then it would lessen their dependence on another person and would benefit them greatly. An extra feature can also be useful where a light/fan turn on automatically when a person enters a room or his/ her presence is sensed inside a room.

Again, if we consider the power consumption at home, power is often wasted when electrical fixtures are left running even when no one is present inside the room. A system can be implemented where the fixtures (i.e. lights, fans) turn off automatically after it senses the absence of any person inside a room.

Many apartment houses have large parking facilities for cars in the basement. There, the parking spaces allocated for tenants are pre-marked or booked, but in large shopping complexes, hospitals or hotels, the spaces are not pre-marked, and parking in these facilities is time consuming if someone doesn't take note of the number and place of the empty parking slots. A system can be implemented where it will easy to monitor the current number

of vehicles in a garage, the number and position of the empty slots etc. It will greatly reduce the complexity of managing a large parking facility.

1.2 Home Monitoring and Control System

Microcontroller based smart home monitoring and controlling system is a system where microcontrollers are used for control and monitor lights, fans, windows, doors and garage through computer software.

The main entrance of home is password protected. A keypad is placed just in front of the main entrance. If anyone presses the correct password then the main entrance will opens otherwise it will remain closed. If anyone presses the wrong password more than three times consecutively, the main entrance will automatically lock and a buzzer will turns on. Without the owner's master password the main entrance will not open. We also monitor the room temperature and set fire alarm. There are options to switch (on or off) lights and fans with remote control.

For a large garage we cannot decide where we will get a free slot to park the vehicle. A mapping board in front of the garage entrance will display the free slots, present number of vehicles and booked slots. When vehicle comes to enter into the garage then the gate will automatically open and when vehicle go out of garage then gate will again open for exiting vehicle.

1.3 Features of the Proposed System

Considering all these problems, we have developed a control and monitoring system for the whole house which has the following features:

- Password protected electronic lock at the main entrance for security,
- An alarm for fire hazard protection,
- The ability to turn on/ off any electrical fixtures and the window using remote control,
- A control and monitor mechanism to turn on the fixtures when a person enters/ remains in a room and to turn these off when no one is present in the room,
- A monitoring mechanism for a large garage management.
- A computer based software for monitoring and control of the whole system has been developed to monitor the fixtures (i.e. lights, fans and window) of the

room and the window. It also monitors the room temperature to determine fire hazards.

- The garage information is informed through another software, where we monitor free slots of the garage/parking lot, the booked slots and the number of vehicles parked in garage. Software also performs for slot booking and control parking gate entrance.

Chapter 2
Working Principle

Our project is designed into three major parts. One part is security system which have password based main entrance. Another one is home fixture monitoring & control, where we can control and monitor room & fire alarm fixture (i.e. lights, fans and window). The last part is garage monitoring & control system. This is designed as to monitor the parking lot condition and control over computer.

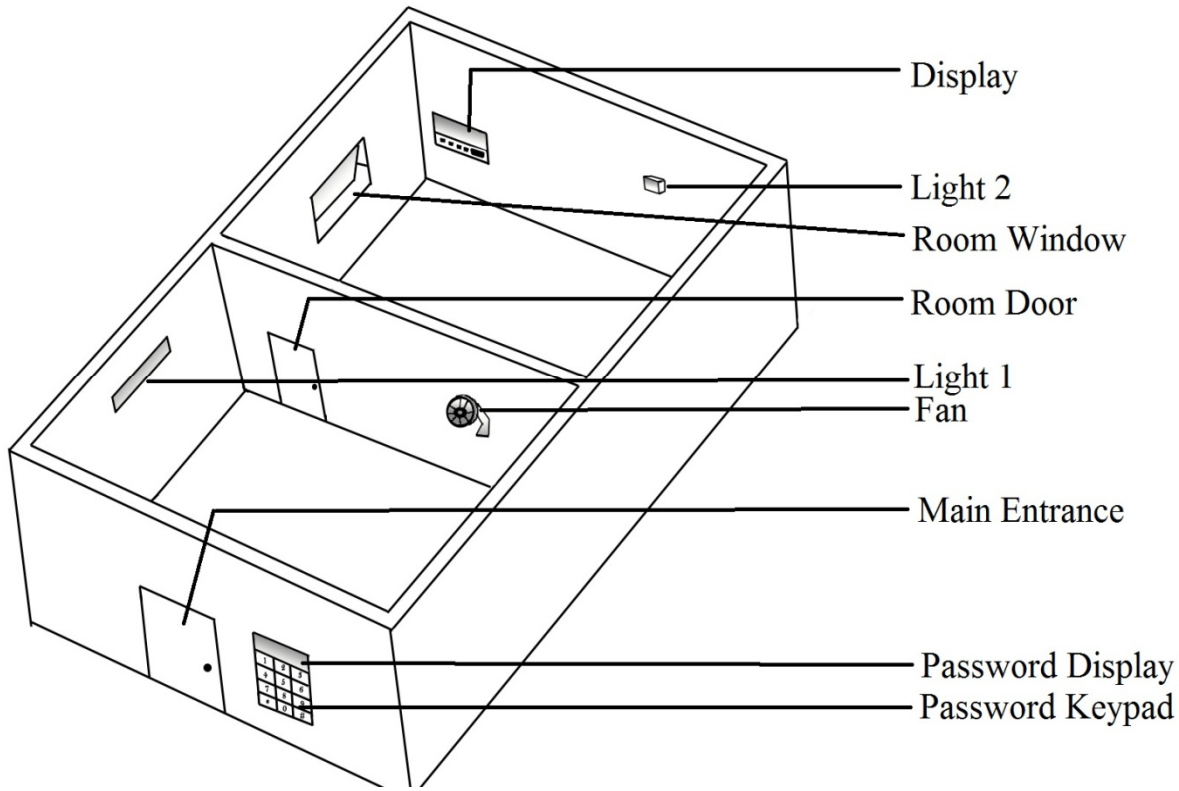

Figure 2.1: Example of Home Monitoring & Control System

2.1 Security System

Security system includes password based entrance security and fire alarm security system. Those are described below.

2.1.1 Password Based Entrance System

The main entrance is locked with password based security system. If anyone tries to enter the home, she/he needs to provide a password. Without the right password the door will

not open. If anyone provides wrong password consecutively for more than three times, an alarm will turn on and then the main entrance cannot be open without the owner's 'master clear' password. Unauthorized entrance blocked through this password protection system. Users have access to change the password.

2.1.2 Fire Alarm System

In this project a room temperature monitoring system is installed. The current room temperature is shown on the software and an LCD display inside the room. If the temperature increases above the user set temperature, fire alarm will turn on. Thus we can minimize damage due to accidental fire.

2.2 Home Fixture Monitoring & Controlling System

Home fixture monitoring and control system includes control of home lights, fans and windows. It also includes counting the number of person inside the room.

2.2.1 Home Lights, Fans Monitoring & Controlling via Software

There is a software based controlling and monitoring system of home fixture (i.e. lights, fans etc). We have developed software named as 'Smart Home' which will monitor and control lights, fans on/off condition and window open/close condition. Here we also have installed timer based power calculation system. We can know how much power is being consumed in the system from this software. We have access to change alarm temperature and observe room temperature though this software.

2.2.2 Auto Window Control System

This system is developed to control the room window. A rain sensor is placed in window to sense rain. If the rain sensor senses rain, the window will close. This window can also be opened or closed by remote control or software.

2.2.3 Room User Monitoring System

In our project we have installed a sensor based room user counting option. Infrared sensors are placed in front of room gate for counting the number of person entering or exiting the room. If there is no user in room, all lights and fans will automatically turn off.

2.2.4 Timer Based Power Calculation System

There is a timer based power calculation system in this project. This power calculation shows how much power is consumed by fixtures. We monitor this value through LCD display or software.

2.3 Garage Monitoring and Control System

Garage monitoring and control system is done through software named as 'Garage'. This system includes sensor based auto garage gate and vehicle position monitoring.

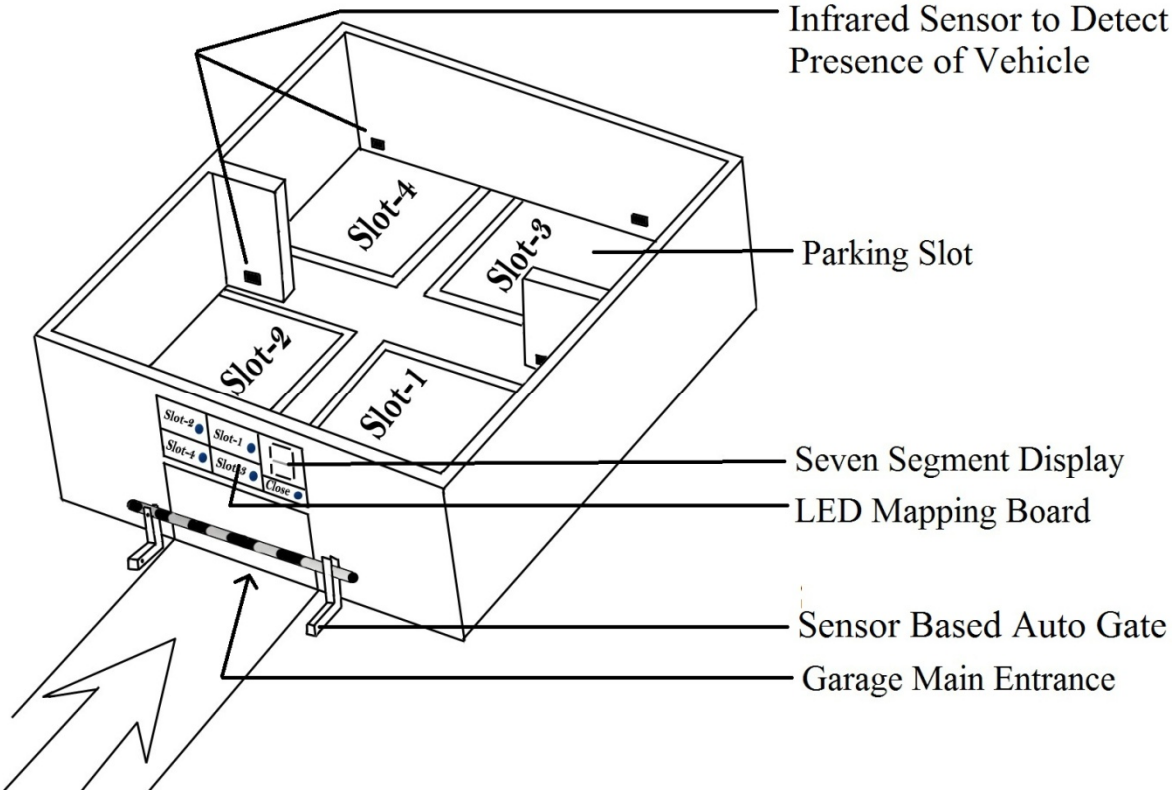

Figure 2.2: Example of Garage with Parking Monitoring and Control System.

2.3.1 Garage Monitoring and Controlling via Software

We attain information about which slot is free and which slot is filled, and the number of vehicle in garage from this software. We control the gate through three modes of operation (i.e. close, open or normal). When in 'close' mode, the gate is closed and stops the sensor auto responding where car cannot enter. When we select 'open' mode, parking system gate is opened and at 'normal' mode, gate will open and close by sensing incoming and outgoing vehicles. We can also book free slots in advance by this software.

2.3.2 Sensor Based Parking Gate Response

In normal mode garage gate is controlled by infrared sensor. One pair of sensors is placed in front of the garage gate and another pair is placed just after the garage gate. When a vehicle comes toward the garage, the first pair will sense the vehicle and open the gate. When the vehicle passes the second sensors pair, the gate close automatically. The same thing happens when the vehicle exits from the garage. This time second pair will sense the vehicle first and open the garage gate to allow the vehicle to exit of vehicle from garage.

2.3.3 Sensor Based Vehicle Monitoring

When a vehicle is parked in a slot, one pair of infrared sensors detects it and displays on the mapping board and also the software. The infrared sensor continuously senses this position until the slot is empty.

Chapter 3
Description of Hardware

In this project 'microcontroller based smart home monitoring and control system' we use some major components such as:

A. Microcontroller
   1. Interrupt
   2. Timer
   3. Comparator
   4. ADC
   5. EEPROM
B. Communication modules
   1. USB
   2. Remote control
C. Sensor
   1. Infrared Sensor
   2. Temperature Sensor
D. Actuators
   1. DC motor
E. Display
   1. LED
   2. LCD
   3. Seven Segment Display
F. Passive Element
   1. Resistor
   2. Capacitor

Those components are described briefly in this section.

3.1 Microcontroller

The main component in our system is the microcontroller. Here we have used PIC18F4550 microcontroller [1] which is manufactured by Microchip Corporation. It has many desirable features such as 2048 bytes ROM (Read only memory), 256 bytes EEPROM (Electrically Erasable Programmable Read Only Memory), a built in USB transceiver [2] for computer interfacing and several 8 bit and 16 bit timers for measuring frequency.

Features of PIC18F4550:

- 8 bit microcontroller
- Flash type 32 KB program memory
- 256 bytes data EEPROM and 2048 bytes RAM
- Total 40 pins among that 32 I/O pins are bidirectional
- On board USB transceiver with on-chip voltage regulator
- One 8 bit and three 16 bit timers, two comparators
- USB interfacing
- Temperature range -40°C to 85°C and operating voltage 2 to 5.5V

Figure 3.1 shows the pin diagram of the microcontroller. There are pins dedicated for different work like ADC, Comparator, Interface, Interrupt etc.

PIC18F4550 Pin Diagram:

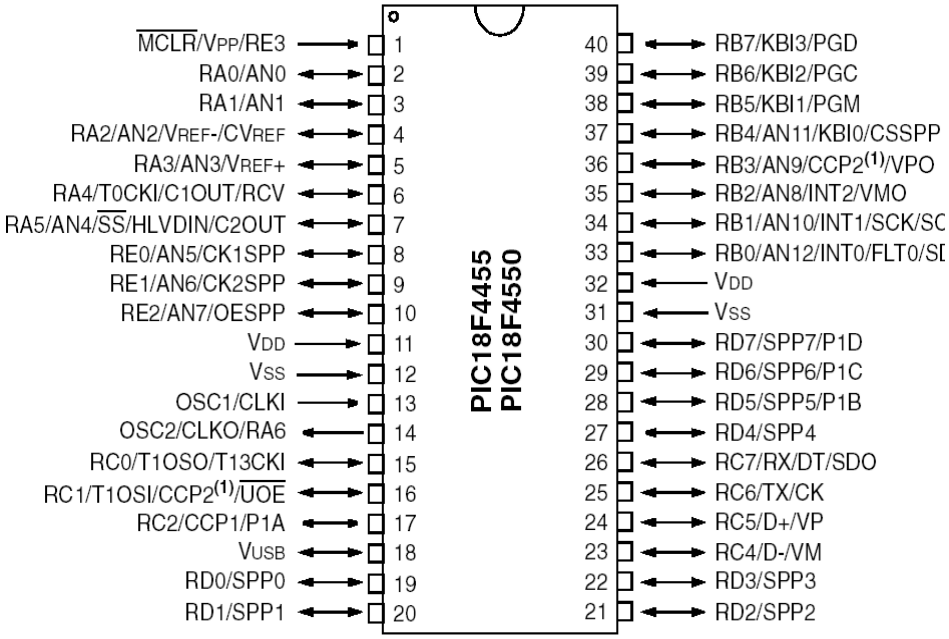

Figure 3.1: Pin diagram of PIC18F4550 microcontroller

In this projects we use some specific features of microcontroller, i.e. interrupt, timer, comparator and analog to digital converter (ADC) features. Those features are described below.

3.1.1 Interrupt

Interrupt is a signal that indicates immediate attention to another event while one event is running. This interrupt can be software interrupt or hardware interrupt. Software interrupt happens when exceptional things are done and is a kind of function call. For example, if we try to divide something by zero then software cannot get any result that time an error will occur. This error comes from sub routine function call.

Hardware interruption occurs when we change a hardware condition such as, by pressing keyboard or moving mouse pointer etc. According to order processor will select which work give priority. Hardware interrupt may run successfully while continuing executes the main instruction [1].

In our project we use interrupt for several hardware and software interrupt such as, USB connection interrupt and timer interrupt as software interrupt and comparator interrupt and pin interrupt are hardware interrupt. The output of the infrared sensor to detect remote signals is also used as external pin interrupt. When we control our system via software then USB interrupt occur. When we press the remote button then the infrared sensor output is taken as an interrupt and we will be counting a time interval through a timer which also gives interrupt on its overflow.

3.1.2 Timer

Timer is just a time counter in a program. We use timer operation as 8 bit or 16 bit in microcontroller. In our project we use timer for controlling garage gate. When the gate is opened then the gate bar will wait for a few moments and then close. The amount of time before closing is controlled though a timer. When calculating the power consumption in a room we use timer of one second duration. We also used timer for remote command receiving to validate the pulse detection.

3.1.3 Comparator

Comparator is used to compare one thing with other things. By comparing some result with each other we can select one result as our desired result [1]. The outputs of the infrared sensor are taken to the microcontroller through the comparator input pin. Here in our project

we use comparator for giving information from sensor to count the member of room users, for vehicle detection and for, automatic door or vehicle parking gate response. Comparator makes high voltage for microcontroller from IR sensor low voltage output.

3.1.4 ADC

ADC is simply an Analog to Digital Converter. It converts analog quantity such like voltages to digital quantity. After this conversion we will get voltages as a binary number. In our project, we use ADC with a temperature sensor to provide temperature output as °C (Degree-Celsius). The Temperature output is a voltage which is proportional of the temperature. This voltage is converted to a digital value sing the ADC. This value is then used in the microcontroller o display the temperature.

3.1.5 EEPROM

EEPROM stands for Electrically Erasable Programmable Read-Only Memory. It is used to store small amount of data. If a data is saved to a microcontroller EEPROM, the data can be read even after the power fall.

3.2 Communication Module

There are two types of communication modules in our system. One is a USB and the other is remote control those are described below. We can maintain and control the whole system by a two types of communication module.

3.2.1 USB

Universal Serial Bus (USB) [3] is used for communication between the hardware (i.e. microcontroller) and personal computer. The USB is plug and play system and is used in many peripherals (e.g. mouse, keyboard etc.) for interfacing. Its price is very low. Here we use USB because it is faster than other communication module such as USART, I2C etc. The USB transfer rate is up to 12 Mega bits per second, whereas USART transfer rate is only 9.2 Kbits. Figure 3.2 shows an example of USB. PIC18F4550 device has a full-speed and low-speed compatible USB Serial Interface Engine (SIE) that allows fast communication between any USB host and the PIC18F4550 [1]. The SIE can be interfaced directly to the USB utilizing the internal transceiver. It has 1 Kbyte USB RAM for uninterrupted data transfer.

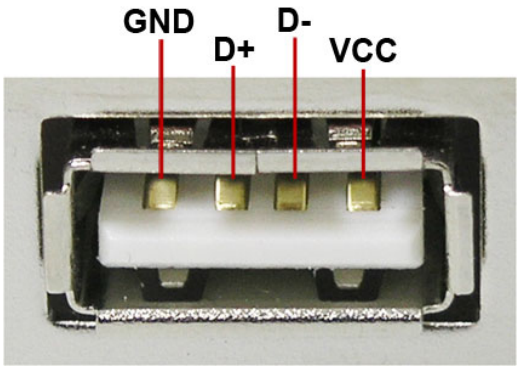

Figure 3.2: Universal Serial Bus

3.2.2 Remote Control System

The remote control system is run by infrared sensor. Different manufacturers follow different protocols. Here we used a SONY HUAYU RM-001A [4] model remote control. SONY follows SIRC [5] (Sony Infrared Remote Control) protocol. Its modulation center frequency is 40 kHz.

Its other features are [1]:

- 12-bit, 15-bit and 20-bit versions of the protocol exist
- For 12-bit protocol; 5-bit address and 7-bit command length
- Pulse width modulation
- Carrier frequency of 40kHz
- Bit time of 1.2ms or 0.6ms

Here in our project, we use 12-bit protocol. The table below lists different address and command sent by Sony remote controls in the 12-bit protocol. Figure 3.3 is shows an example of Sony remote key 19 signal.

Here in our project we use remote control address 1 which is for TV devices. Here we used key 1 for displaying power, key 2 for window on and off, while key 3, 4 and 5 are for displaying light 1, 2 and 3 power. Keys 7, 8 and 9 are for switching on or off lights 1, 2 and 3 while keys 16 and 17 are for adjusting alarm values (temperature). Finally keys 18 and 19 are for adjusting user in room.

Table 1 Remote Control Address Device

Address	Device
1	TV
2	VCR 1
3	VCR 2
6	Laser Disc Unit
12	Surround Sound
16	Cassette deck / Tuner
17	CD Player
18	Equalizer

Table 2 Remote Control Command Function

Command	Function
0	Digit key 1
1	Digit key 2
2	Digit key 3
3	Digit key 4
4	Digit key 5
5	Digit key 6
6	Digit key 7
7	Digit key 8
8	Digit key 9
9	Digit key 0

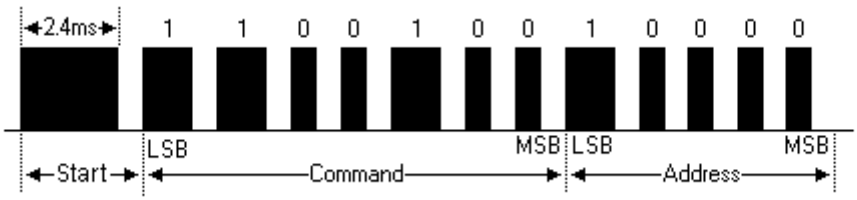

Figure 3.3: Example Signal Diagram for Key 19 of Sony TV Remote

In figure 3.3 is an example of volume- key of Sony TV remote. Here bit is 2.4 ms long, which is starting bit. Than the 7 bit is for command and the last 5 bit is for address.

3.3 Sensors

We use two types of sensors in this system. One is the infrared sensor and another is the temperature sensor. Infrared sensors are used for sensing vehicles in the garage and human presence inside the room. The temperature sensor is used for sensing room temperature.

3.3.1 Infrared Sensor

IR sensor [6] is simply a diode which is sensitive to infrared radiation. It consists of infrared transmitter and infrared receiver. Infrared waves are invisible to the human eyes and its range is 0.75 µm to 6µm [6]. The working principle of IR sensor is based on the change in resistance of the IR receiver. If there is no IR light then a high resistance in MΩ range will be placed in the IR receiver. When IR light is on the IR receiver then resistance will come down

KΩ to Ω range. Then this resistance is converted into voltage change and compared with a threshold voltage. If the voltage of the sensor is more than the threshold then the output is high and otherwise it is low [6].

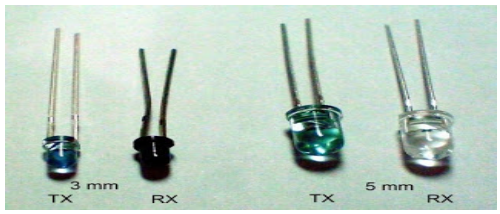

Figure 3.4: IR sensor- Transmitter (TX) and Receiver (RX)

IR sensor comes in two packages of 3mm or 5mm diameter, the 3mm diameter is better for our project because it takes less space.

Infrared transmitter and receiver:

If the transmitter and the receiver diodes have the same color, then we need to test with a multimeter to distinguish them. The receiver diode always gives a higher resistance when there is no infrared light.

Figure 3.5: IR receiver (RX) is connected with multimeter

Here in Figure 3.5 IR receiver (RX) gives higher resistance. In case of IR transmitter (TX) multimeter will not give any display.

3.3.2 Temperature Sensor

The temperature sensor [7] is used to sense the ambient temperature. Here in our project we use LM35 as our temperature sensor. Its output voltage is linearly proportional to the temperature. There are two operating mode of LM35 sensor [7]. We use the operating mode as shown in Figure 3.6. This mode starts temperature counting and displaying from 0°C (Degree-Celsius). Other mode starts temperature counting and displaying from -55°C (Degree-Celsius). We use the first mode as our operating mode in our project.

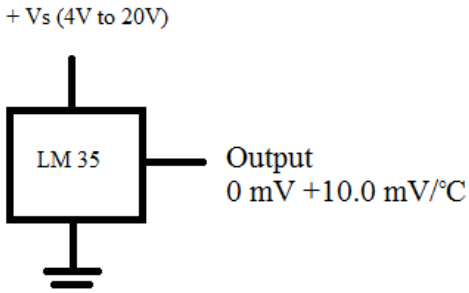

Figure 3.6: Operating Mode of LM35 Temperature Sensor

3.4 DC Motor

In our project we use DC motor to control home door, window and garage main gate movement. An electric motor is a machine which converts electrical energy into mechanical energy. Electric motors are classified into two different categories: DC (Direct Current) and AC (Alternating Current). DC motor is operated by DC voltage.

As we know, DC motors [8] require high driving current. To supply this high current, we use DC motor drivers. DC motor driver simply works as current amplifier and takes low current control signal and gives higher current signal. Here in our project we use DC motor drivers model L293D. One DC motor driver can drive two DC motors.

Table 3 shows the pin description of DC motor driver [9].

Table 3: Pin Description of DC Motor Driver

Pin No	Function	Name
1	Enable pin for Motor 1; active high	Enable 1,2
2	Input 1 for Motor 1	Input 1
3	Output 1 for Motor 1	Output 1
4	Ground (0V)	Ground
5	Ground (0V)	Ground
6	Output 2 for Motor 1	Output 2
7	Input 2 for Motor 1	Input 2
8	Supply voltage for Motors; 9-12V (up to 36V)	$V_{CC\ 2}$
9	Enable pin for Motor 2; active high	Enable 3,4
10	Input 1 for Motor 1	Input 3
11	Output 1 for Motor 1	Output 3
12	Ground (0V)	Ground
13	Ground (0V)	Ground
14	Output 2 for Motor 1	Output 4
15	Input2 for Motor 1	Input 4
16	Supply voltage; 5V (up to 36V)	$V_{CC\ 1}$

3.5 Display

Display is a very important component to show output directly. We can easily show our circuit result by displaying output.

3.5.1 LED

LED is short form of light emitting diode. It is a semiconductor lighting device. LED is just a diode consisting of anode and cathode. When LED is forward biased that means it is switched on and then electron and holes recombine and energy releases as photons.

LEDs are mainly used as visual indicators of different electrical circuit. We sometimes use LEDs for displaying output. Nowadays LEDs are used as lighting devices instead of incandescent and fluorescent lights. The LED has high longevity and is small in size to handle.

Depending on current and heats generation its lifetime is around 2500 hours to 1000000 hours. According to the color and size there are many types of LED. Every LED has its own voltage levels such like: Red LED: 1.63~2.03 V; Blue LED: 2.48~ 3.7 V; Green LED: 1.9~4 V; Ultra Violet LED: 3.1~4.4 V; White LED: 3.1~4.4 V.

3.5.2 LCD

LCD is a type of display where we can display any kind of numbers, images, graphs etc. LCD is short form of Liquid Crystal Display. It uses liquid crystal light emitting properties and it does not reflect light directly. LCD is a low power device whose power requirement in a range of few microwatts.

Here in our project we use three LCD displays to show vehicle parking information, home system information and password keypad information.

3.5.3 Seven Segment Display

In this project, we use one seven segment display to display the number of vehicles in the garage slot. Although the seven segment display can be directly used with a microcontroller without a driver, but then it requires too many I/O pins and multiplexing technique, so we connect the display to the microcontroller using BCD (binary coded decimal) to seven segment display driver. There are two types of seven segment displays: Common anode and Common cathode Figure 3.7 shows that.

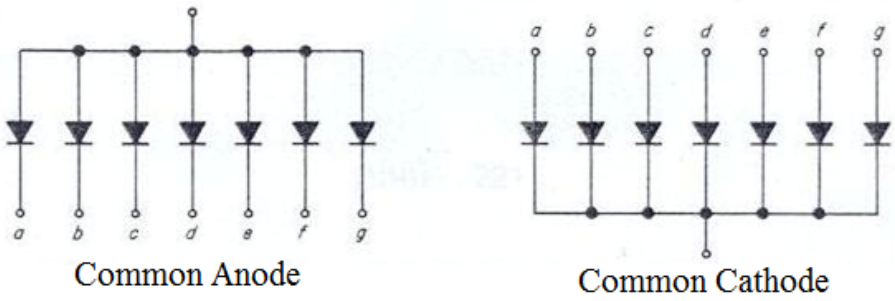

Figure 3.7: Two types of seven segment display

For the common anode type seven segment display we need the 7447 driver and for the common cathode seven segment displays we need to use 7448 driver. In our project, we use a common anode type display and the 7447 driver.

3.6 Comparator (LM 324)

In our system we use a comparator to sense voltage difference in vehicle parking system. Voltage in a free slot and occupied slot are different. We sense this voltage difference and compare it with our reference voltage. According to the reference voltages it sends information to the system whether the slot is free or occupied. We use LM324 [10] as our comparator device.

3.7 Passive Elements

Elements in a circuit which do not produce power are called passive elements. Resistors, capacitors are passive elements in electric circuits.

Resistor

Resistor is a passive component which resists the flow of current in a circuit. A voltage drop always occurs across a resistor. Resistors always follow Ohm's law V=IR where V= voltage, I= Current and R= resistance. Resistor denotes by Ω (Ohm).

In our project, we use 220Ω, 1KΩ, 6.8KΩ, 10KΩ resistors to complete the project.

Capacitors

A capacitor is a passive component which stores energy in an electric field. Capacitors are two types according to build material, ceramic and plastic type. Construction is different but every time needs at least two electrical conductors which are separated by a dielectric. In this project, we use 22pF, 20pF, 1uF capacitors.

3.8 Crystal Oscillator

Crystal oscillator is an oscillator which gives signal of fixed frequency. Here we use 16MHz and 8 MHz crystal to generate signal.

Chapter 4
Software

The system has two types of software. One type of software is of microcontroller programming written in c programming language and another is for the PC (written in Visual C++). Those software systems are described below.

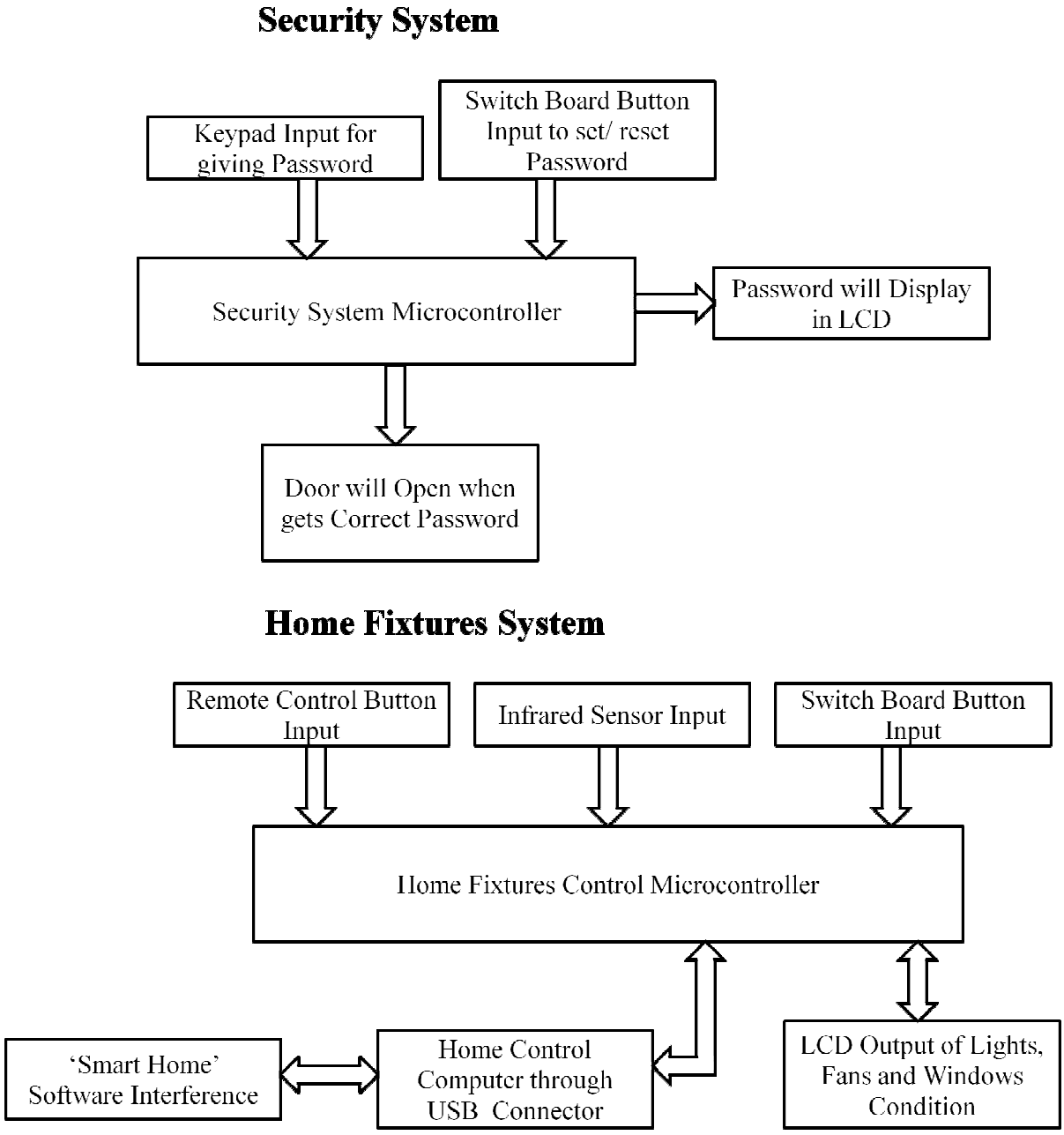

Figure 4.1: Security and home fixtures monitoring and control system

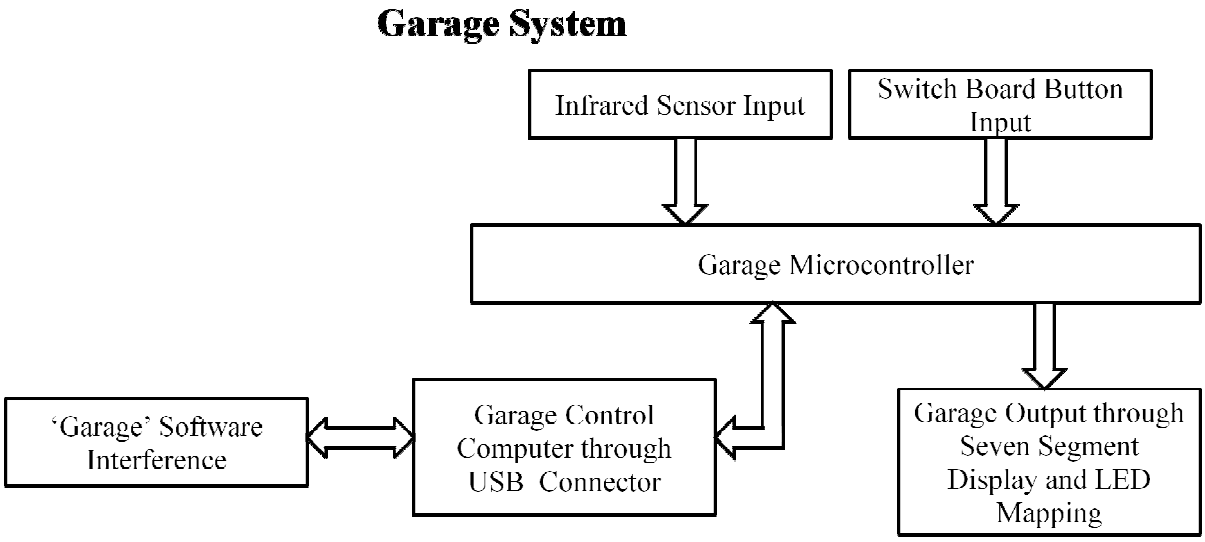

Figure 4.2 Microcontroller based garage monitoring and control system

4.1 Microcontroller Software Details

We designed three programs for the microcontrollers for our project. These are for the security system, the home fixtures control system and the garage system. These software algorithms are in described below.

4.1.1 Security Algorithm

Security algorithm is used in the password based entrance system. The security door includes a keypad, some switches and on LCD display. When we press the password in the keypad which is situated in front of the door, it will show in the LCD display. When the security lock gets the right password, then the main entrance will open. If anyone tries the wrong password consecutively for three times then the security alarm will turn on. Then, except the owner, no one can unlock the door.

Algorithm: Figure 4.3 shows the security system algorithm. At first the main entrance system initializes and the EEPROM will check the system's password. We have two options, one is the main entrance open option and another is the password change option. When we want to change the password we press # and change the password. Again, when we want to open the door we starts with * button. If password is correct, then the door will open. The LCD display will show the password that was already entered through the keypad. Then, the system will check its EEPROM to write again.

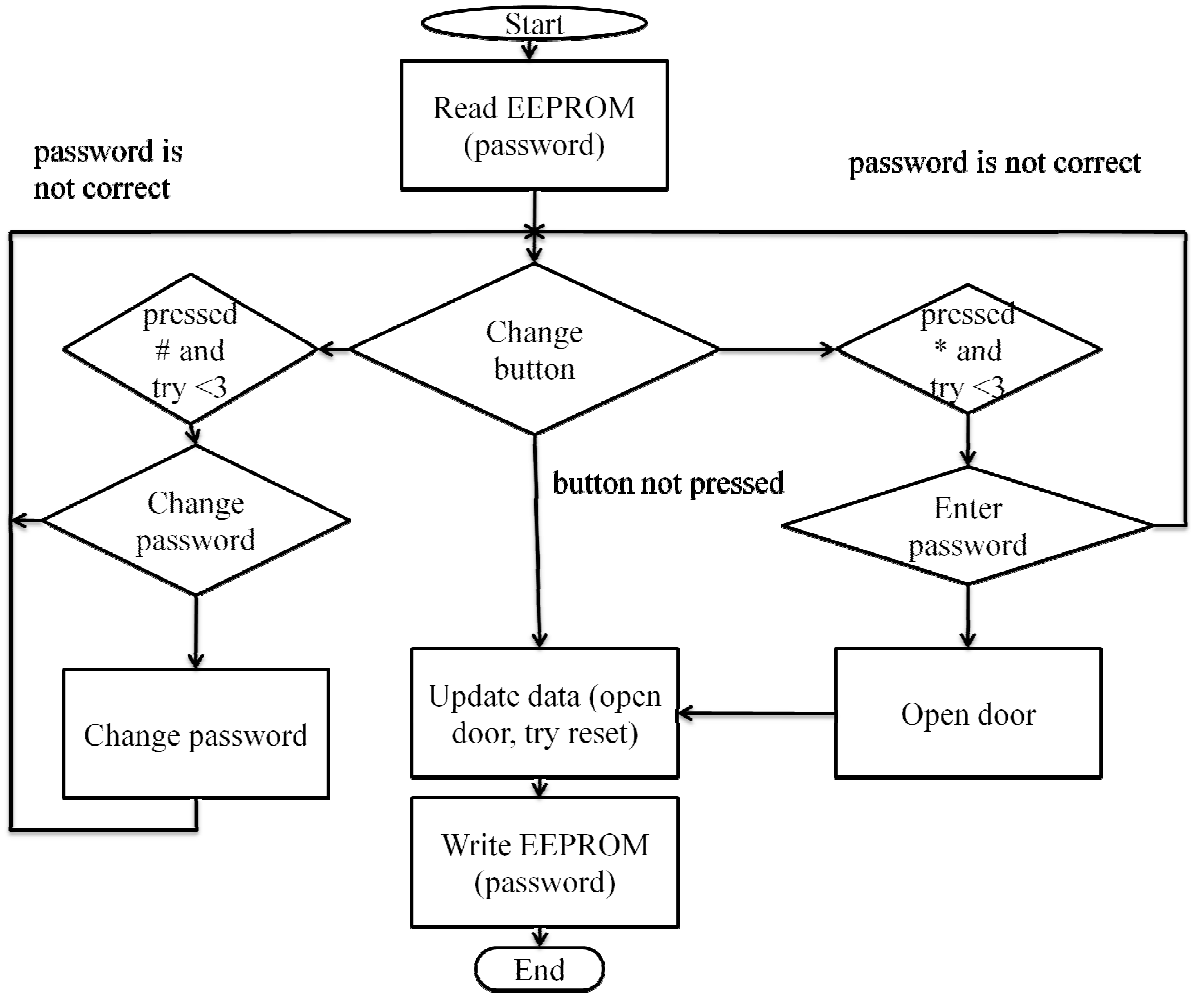

Figure 4.3: Security System Algorithm

4.1.2 Home Fixtures Control System Algorithm

Fixtures control system includes sensors, remote control, button, PIC18F4550 microcontroller. Fixtures control system IC is connected with computer software through USB cable. So, now we can control and monitor home fixtures from computer software.

Algorithm: Figure 4.4 shows the home fixtures control system algorithm. Here at first home system initializes and then EEPROM will check system's memory. Then the USB will be. If there are any changes through the keyboard buttons or the remote control happens then the system updates its information. If no change happens then it will go to the comparator input. Here we use two comparators to count the number of people in a room. According to the comparator information the system will run its COUNT+ or COUNT- option. Then the system will again check the remote or button input or change and update its data of lights, fans and windows condition and the power consumption information. Then LCD will show

the updated result. Then again system will check its EEPROM and USB communication to write.

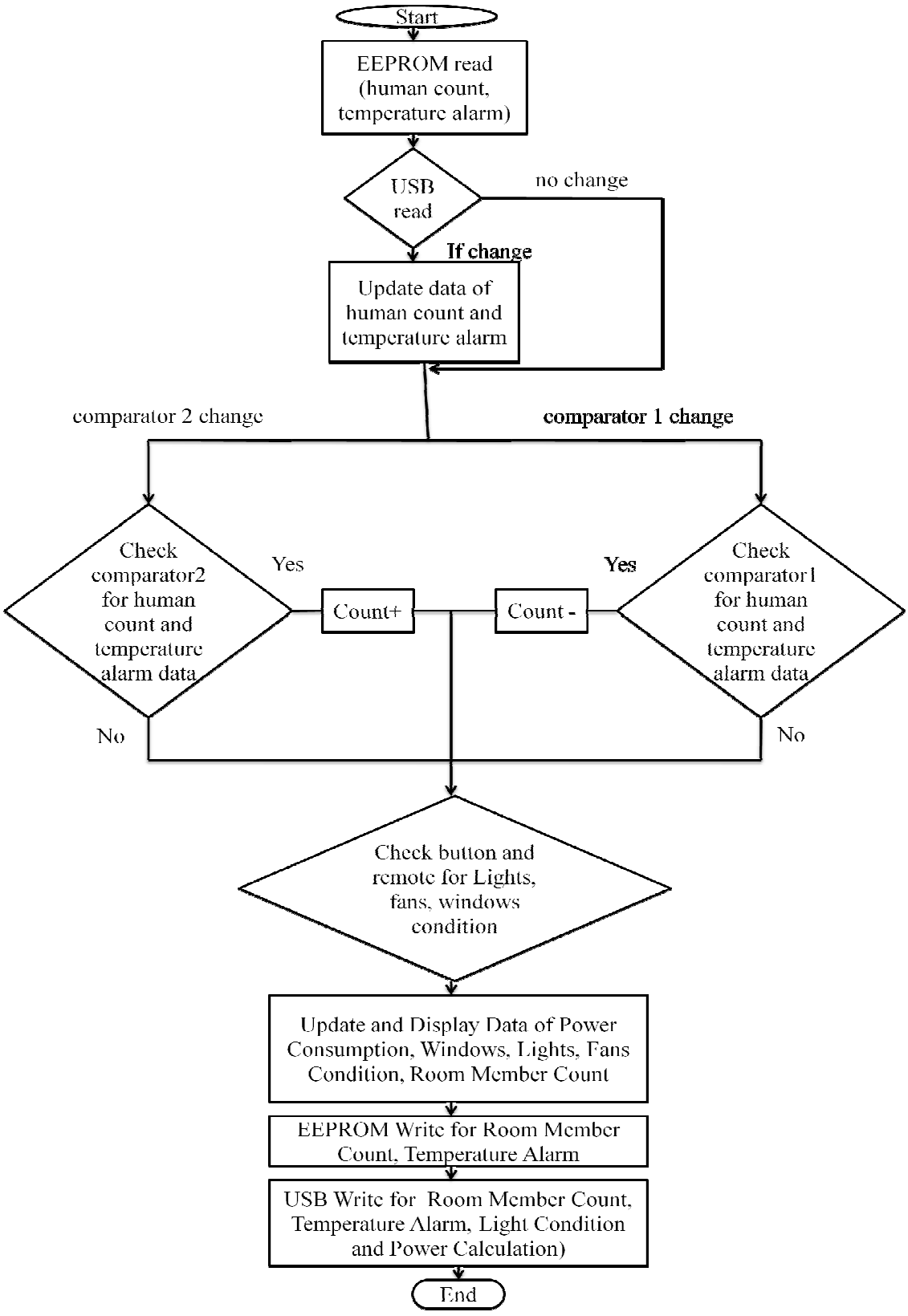

Figure 4.4: Home Fixtures System Algorithm

4.1.3 Garage System Algorithm

Garage system includes infrared sensor, DC motor, buttons and garage output through LED mapping. Infrared sensors are used for sensing ingoing and outgoing of vehicle. It is also used for detecting vehicle position in parking slot. This part is connected with the computer the software via USB cable. So, we can control and monitor the garage system from computer software.

Algorithm: Figure 4.5 shows the garage system algorithm. At first the garage system IC initializes then the EEPROM will check the systems memory. According to the memory, the system will rearrange its mapping and counting. This system is connected with the computer software through USB cable, so the system also reads USB instructions. If there is anything changes shows up, then system will update its value. If no change happens, then the system will check the input button. If there is any change, the system will update its value. If no change happens, then the system will check the sensor input, display its result in LED mapping board and seven segment display. It also checks the USB communication for any software instructions. Then, again system will check its EEPROM and USB communication to write.

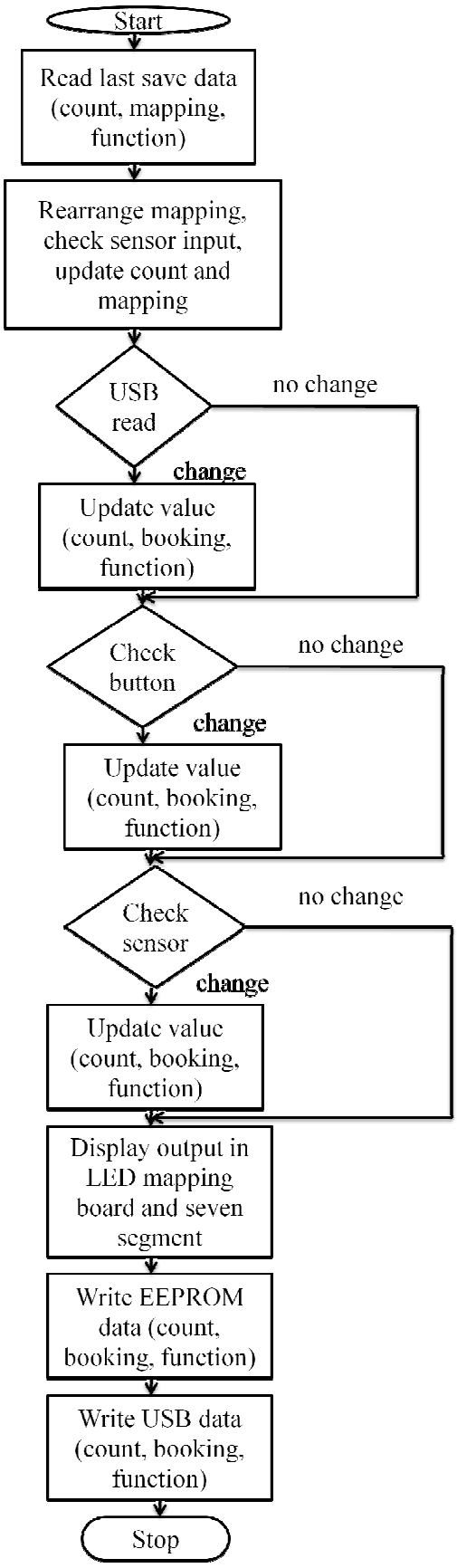

Figure 4.5: Garage System Algorithm

4.2 Computer Interfacing Software

Computer interfacing means establishing communication between computer and user or device. Interfacing may be on bidirectional or unidirectional. Here in our project we developed 'Smart Home' software for controlling home fixtures and 'Garage' software for controlling parking system. We have used Microsoft Visual C++ to build these software.

4.2.1 Smart Home System Control Software

This software is designed to be control and monitor all home fixtures and window as well as the main entrance security and fire alarm. Now we will describe the Graphical User Interface (GUI) software (Figure 4.6).

Light Power Setup Window: In this window, we can preset every light's rating. Here we only show lights but we can also set fans instead of lights.

Monitoring Window: It shows the individual consumption of power for each light.

Temperature Window: It shows the current temperature of the home which is being updated automatically according to the home environment.

Alarm Window: We can set a temperature value for alarm in this window. When the temperature increases above our set temperature value then an alarm will automatically turn on.

Person in Room Window: It shows how many persons are in room. We can manually adjust the number by using count+ or count- button.

Reset: It gives option to reset power consumption of lights and room member counting system.

Window Control: In window option we can control window close or open situation. When window is closed then it shows red color and when window is open it shows green color.

Light Condition and Operation Window: It shows which lights are on and which are off. If light is on then it shows green color and if off then it shows red color.

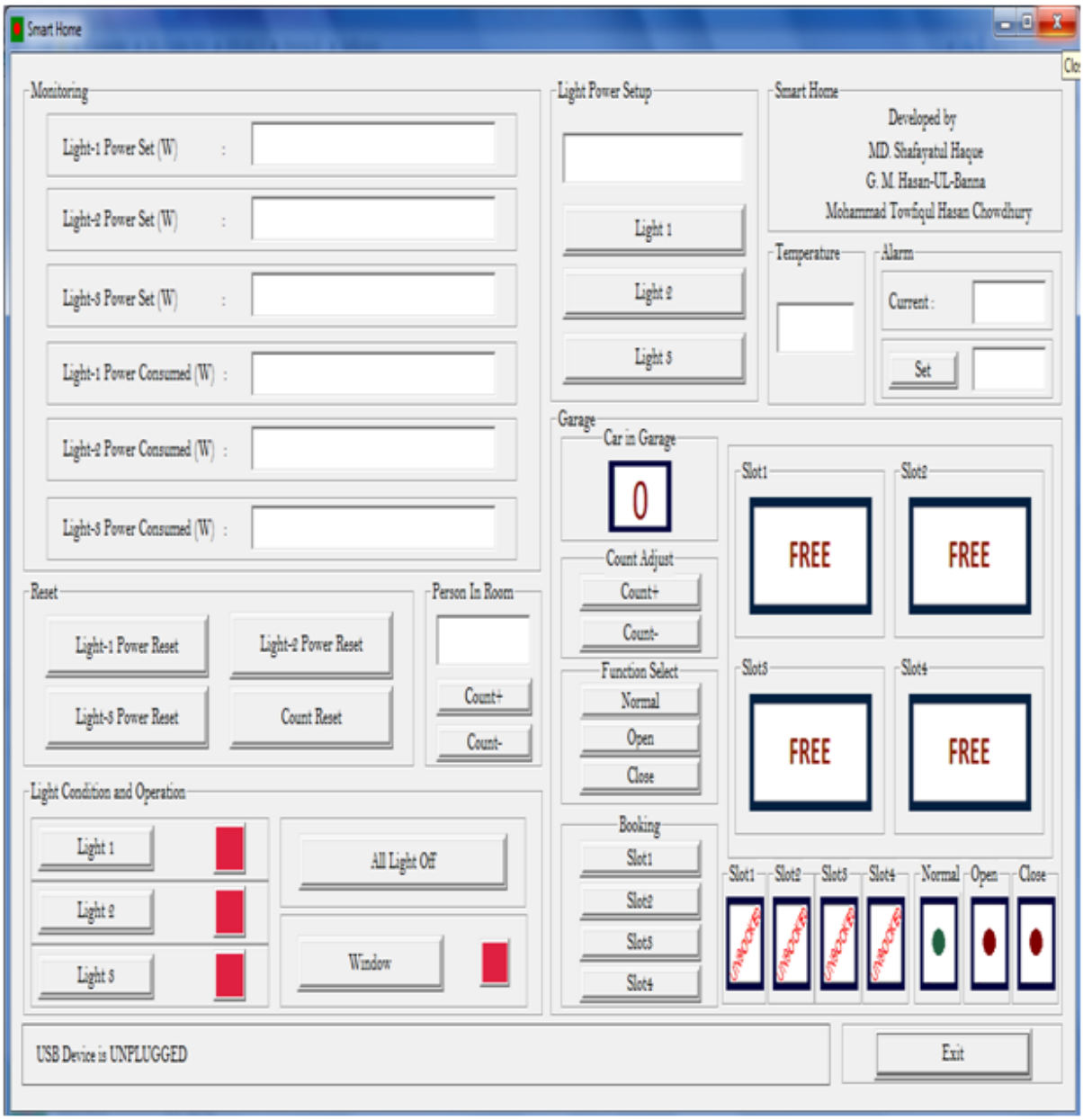

Figure 4.6: GUI of Smart Home System Software Control

When USB is connected between hardware and software then it shows ‘Smart Home Device’ is plugged in and when device is not connected then it shows USB device is unplugged. In Figure 4.6 software is not connected with hardware and for that it displays USB device is unplugged in the bottom left corner. We can close this software interference by clicking exit button option.

4.2.2 Garage System Control Software

This software is designed as it can control and monitor garage system. Figure 4.7 shows the 'Garage' software. We have used Microsoft Visual C++ to build this software. Now we will describe about the Graphical User Interface (GUI) software. This work interface is also incorporated in our 'Smart Home' software. We made this as a separate software. So that the garage can be separately controlled from computer located in the parking slot manager's room.

Car in Garage Window: It displays the total number of vehicle is present in the garage. Our vehicle capacity is maximum four. If it shows minimum zero then its means there are no vehicle in any parking slots.

Count Adjust Window: This window is for maintaining the vehicle count. If there is anything wrong with the vehicle counting subsystem then we can adjust vehicle number manually by this window.

Function Select Window: This window is for controlling garage gate condition. In 'normal' condition, when a vehicle tries to enter or exit, then the gate will open automatically. The function status in the 'normal' button window will give green signal as displayed bellow in Figure 4.7. If the 'open' option is select then gate will always open. The function status window for open button will give green signal. If 'close' option is select then the gate will be always closed. The function status for 'close' button window will now give green signal.

Slot Window: This window is for displaying vehicle. When there are no vehicles then it displays FREE and when a vehicle comes then it display an image of CAR.

Booking Window: In garage parking system there are four slots. To book those four slots we use this window. We can book slots when slots are free. When we book a free slot then it displays as BOOKED.

When USB is connected between hardware and software then it shows 'Garage' is plugged in and when not connected then it shows USB device is unplugged. In Figure 4.7 software is not connected with hardware and for that it displays 'USB device is unplugged'.

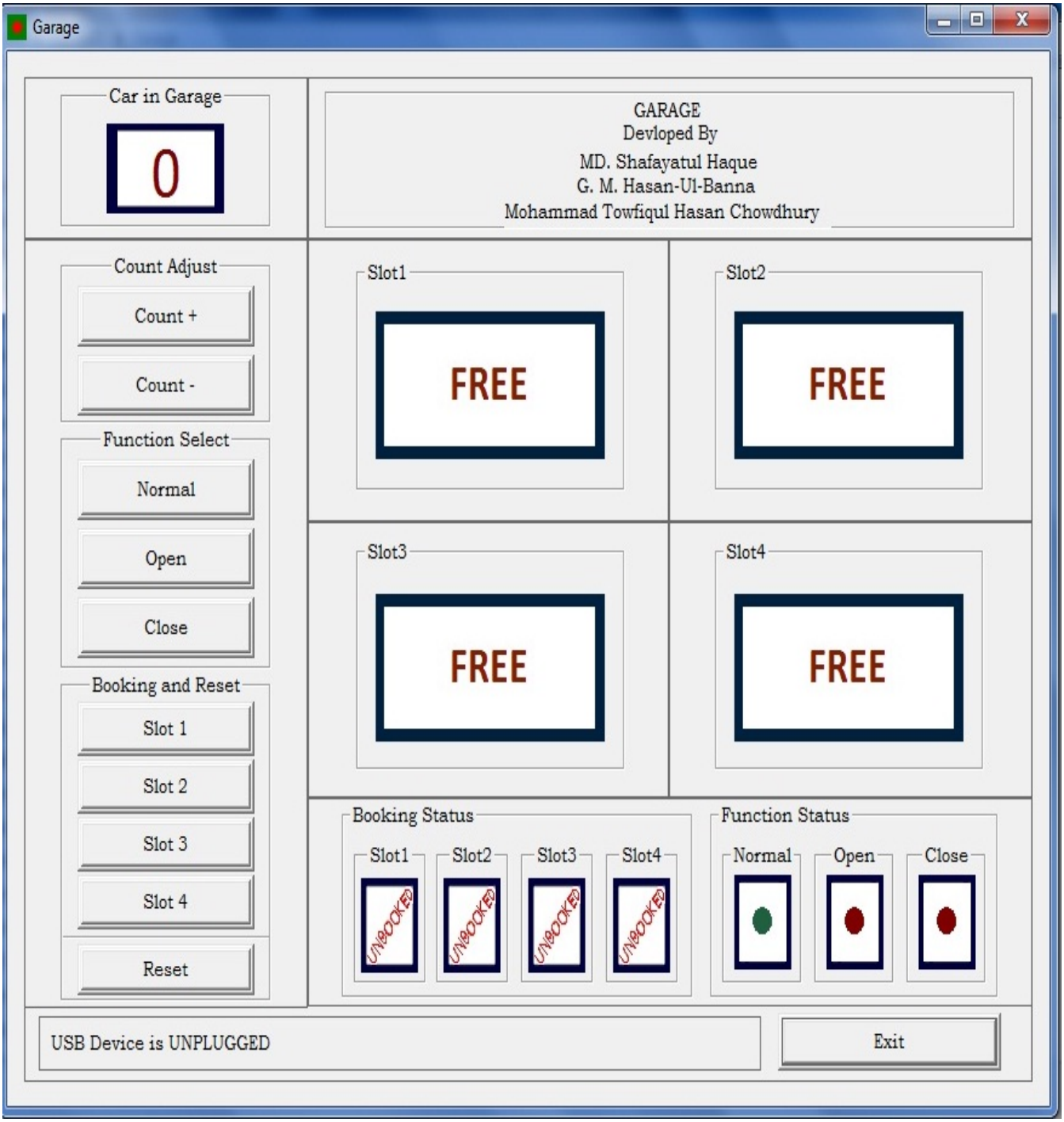

Figure 4.7: GUI of Garage System Control Software

We can exit this software by clicking exit button.

Chapter 5
Hardware construction

Our project's hardware system is divided into three basic parts. These are security part, home fixtures monitoring and control and garage. The block diagram of the whole system is given below.

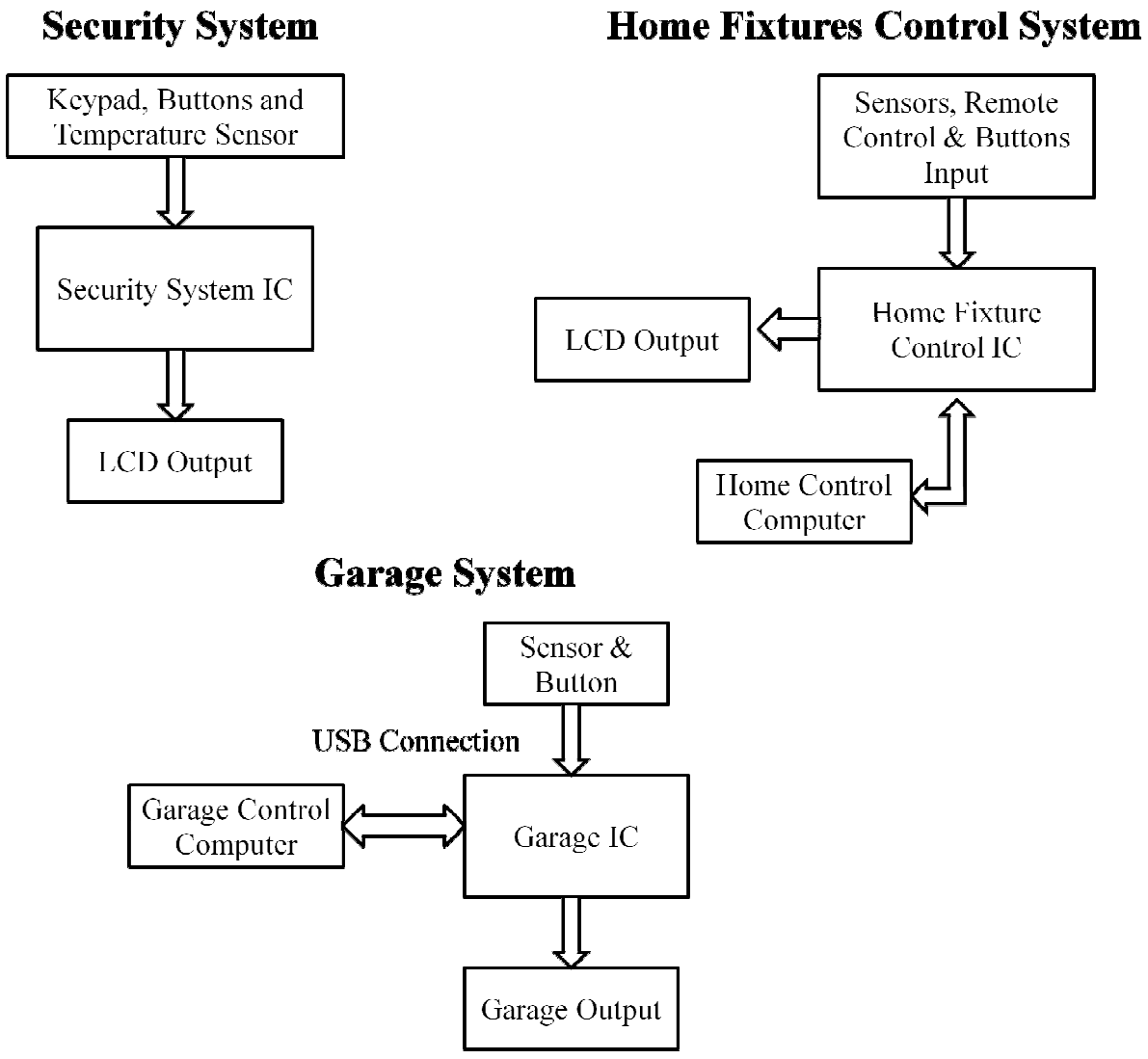

Figure 5.1: Block diagram of System

Now we will discuss the whole hardware system.

5.1 Circuit Construction

Circuit construction of the system is divided into three major parts. These are security circuit, home fixtures control and monitoring circuit and garage system circuit construction. Those are described below.

5.1.1 Security System Hardware

The security system hardware includes the main entrance hardware. Figure 5.2 shows the security system of the main entrance. This circuit is for maintaining the main entrance control IC. For that we use PIC18F4550 microcontroller. USB is used for connecting between hardware and computer software. From this circuit, we can control security entrance. Piezo buzzer is used for security alarm. Security alarm will turn on when any person give wrong password consecutively for more than three times.

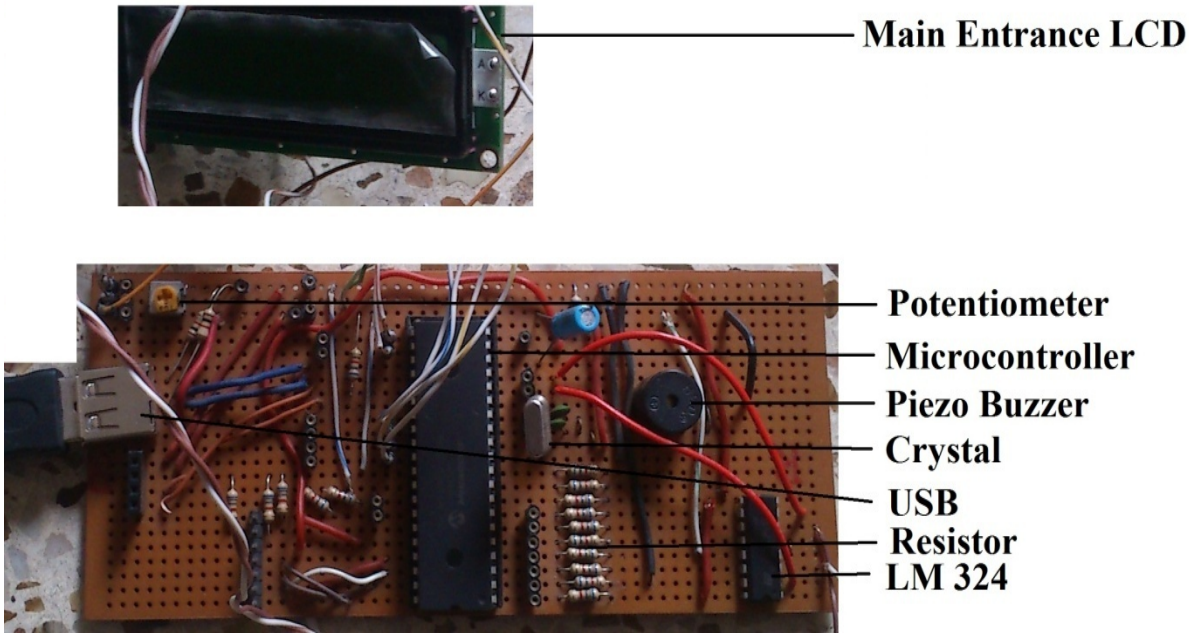

Figure 5.2: Security Control circuit

5.1.2 Home Fixtures Controlling and Monitoring Hardware

Figure 5.3 shows home fixtures controlling and monitoring system hardware. This circuit is for controlling room fans, lights and windows. The temperature sensor LM35 senses the room temperature. A piezo buzzer is connected for fire alarm. If temperature increases above the user set temperature, piezo buzzer will turn on. The LCD displays the room temperature, on/off status of lights and fans and close/open status of windows.

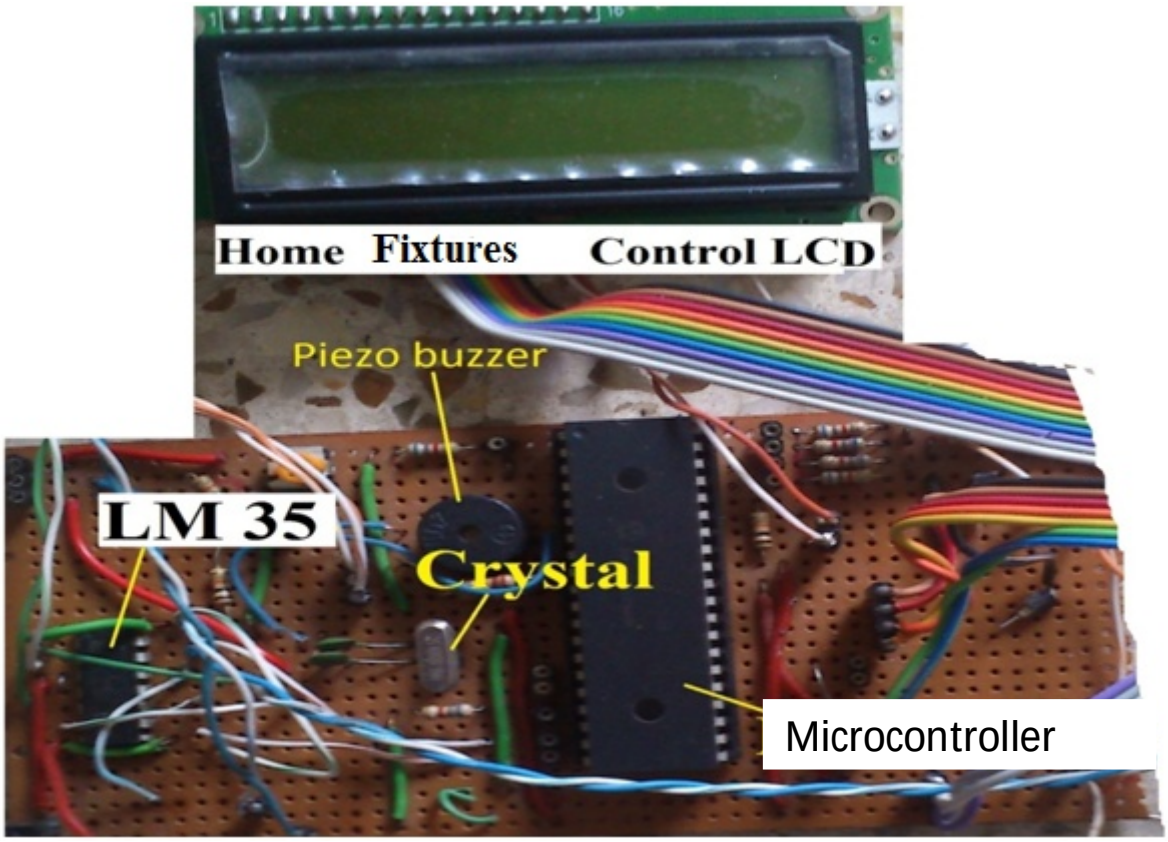

Figure 5.3: Home Fixtures Controlling and Monitoring Circuit

Figure 5.4 shows the schematic diagram of the security system IC and Figure 5.5 shows the schematic diagram of the home fixtures control and monitoring system. We can control room lights, fans and windows by this circuit.

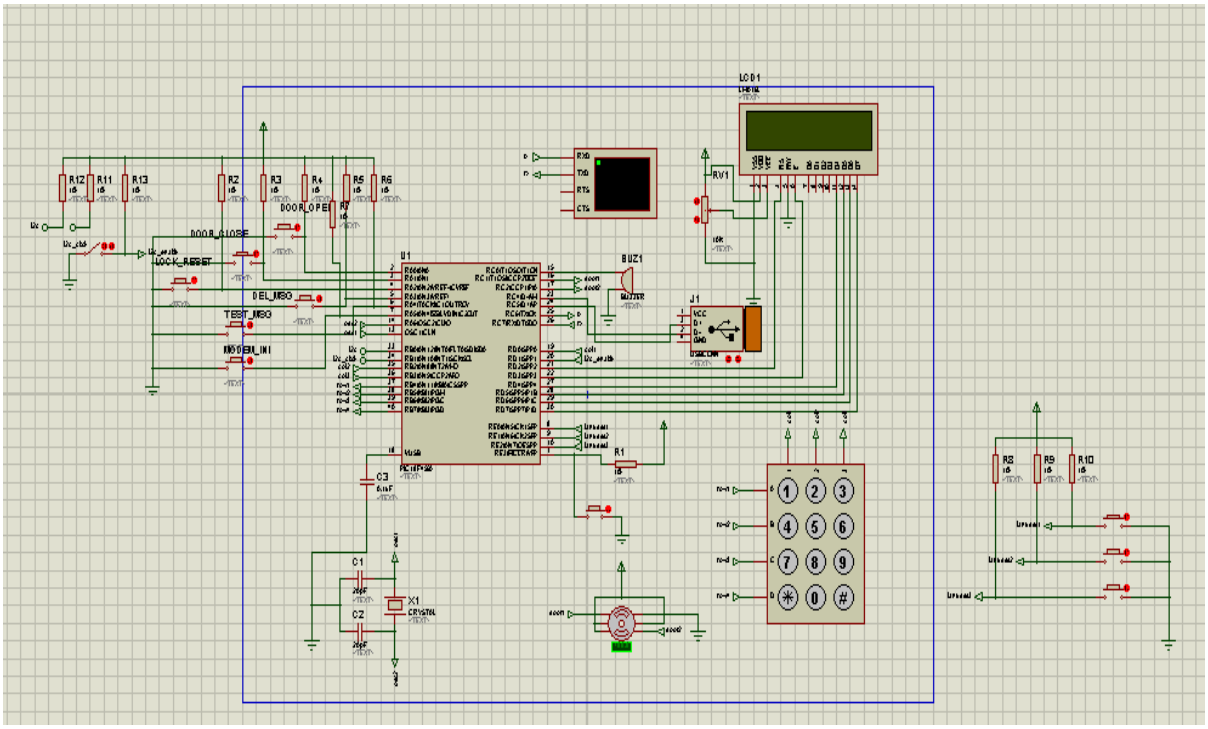

Figure 5.4: Security Control IC schematic

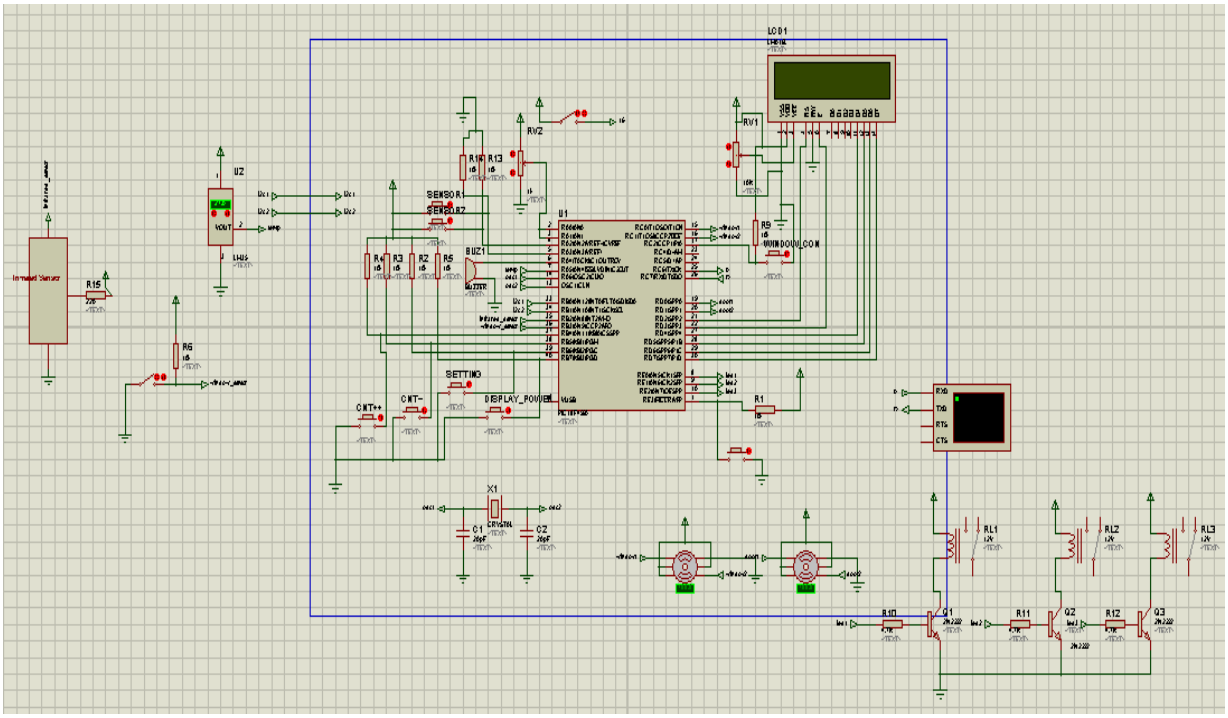

Figure 5.5: Home Fixtures Control and Monitoring IC schematic

5.1.2 Garage System Hardware

Garage system hardware is divided into four major parts. Those are project board, parking slot board, display mapping and switch board. Those are described below.

Project Board:

This is the main circuit board (Figure 5.6) of our garage system. One microcontroller, 8 MHz crystal, two LM324 comparators, some capacitors and an USB female socket are connected into the system. The crystal is used for a clock for the microcontroller. The comparator is for comparing efficient values of voltage. The USB is used for connecting this hardware with computer (software).

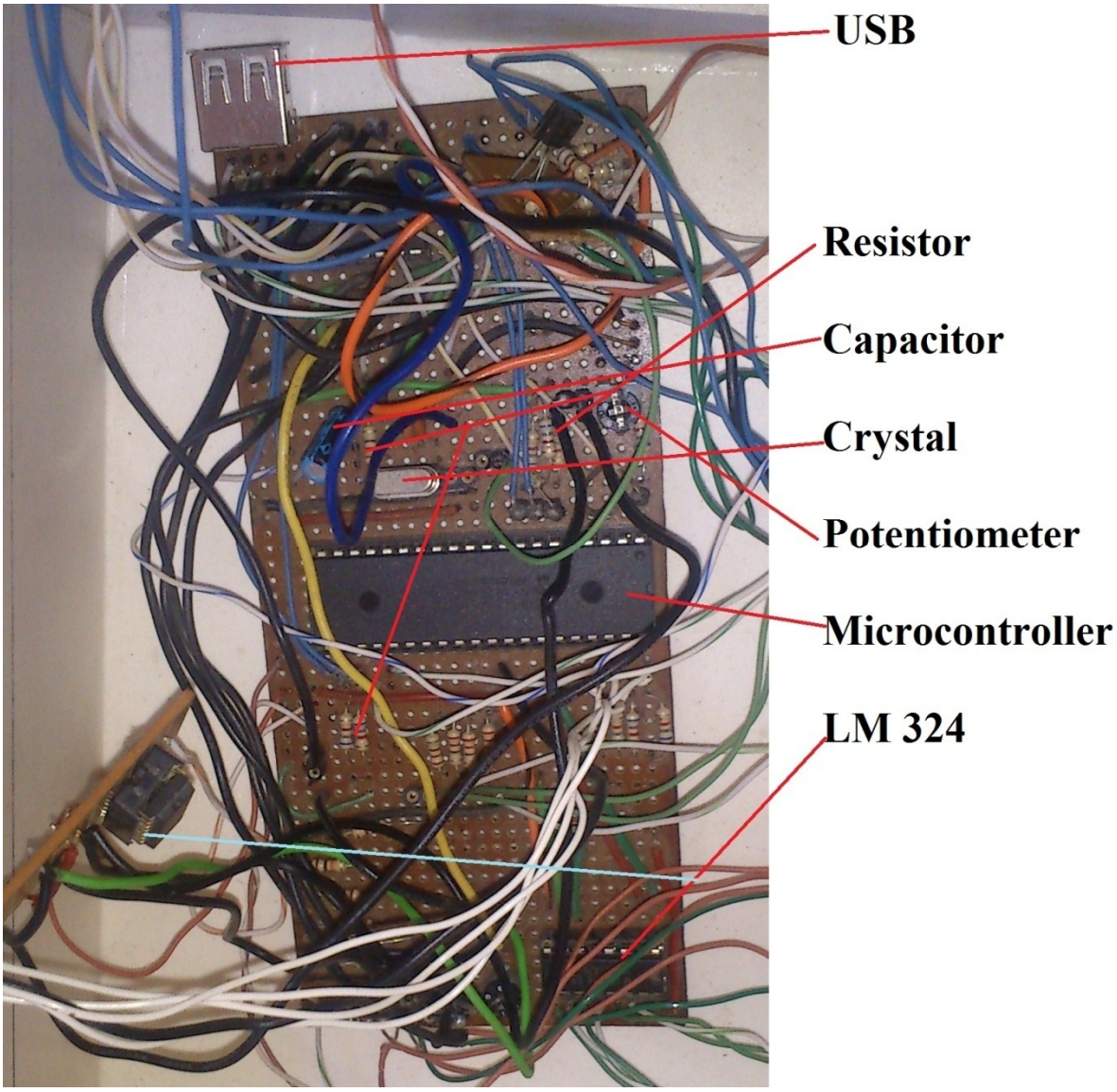

Figure 5.6: Garage System Circuit

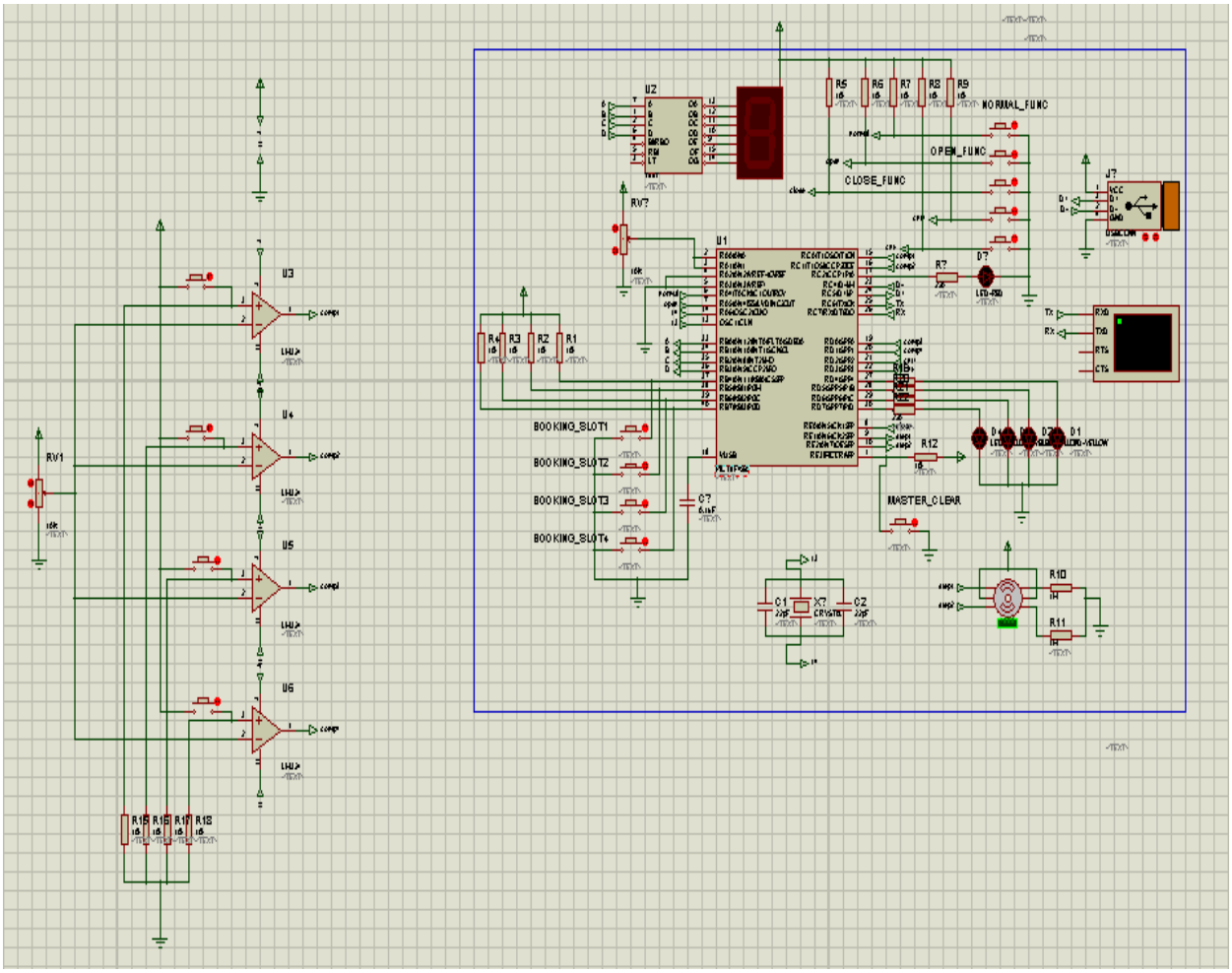

Figure 5.7: Garage System circuit

5.2 Framework Construction

For demonstrating this project we have built a model comprising a ‘house framework’ and a ‘garage framework’. House framework includes home main entrance security and home fixtures control. Garage framework includes garage control and monitoring part. We have used PVC sheet to complete our framework construction.

5.2.1 Home System Framework

The home system framework part consists of the house framework and house switch board framework. Those are briefly described below.

House Framework:

The house framework has one main entrance, one room door and one room window (Figure 5.8). When human goes toward to the room door, the sensor will sense human and the

doors will open automatically. The doors will also close automatically when human exit the room door. DC motors are used to perform this action. This home system has two lights and one fan. Lights and fans are both simulated by LEDs.

Home Switch Board Framework:

Home switch board can control the whole home system including home main entrance security system and home fixtures control system part. Figure 5.9 shows the switch board of the home system. Switch board is divided into two parts. One part is for home main entrance security part and another part is for home fixtures controlling and monitoring part.

Home main entrance security part has six switches and a keypad. The keypad is used for password input. We can also change the password through this keypad. The main home entrance control LCD shows the password which is pressed.

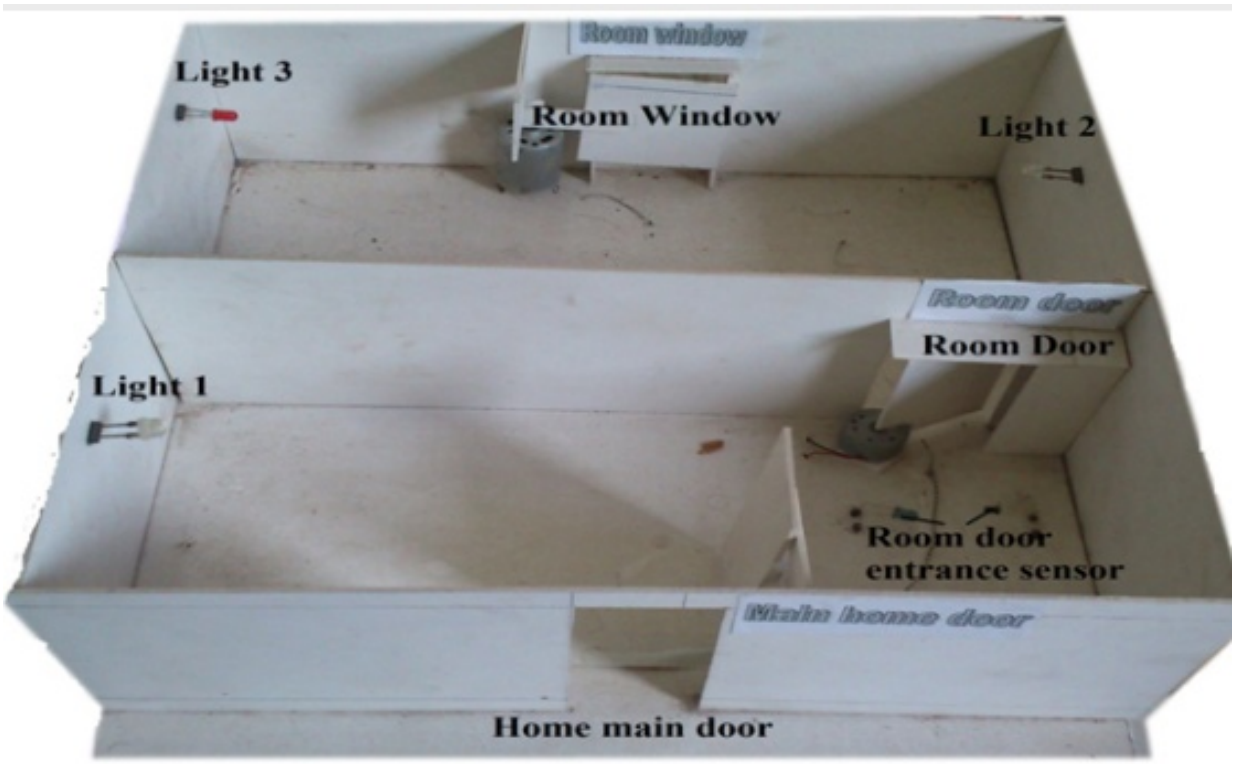

Figure 5.8: House Framework

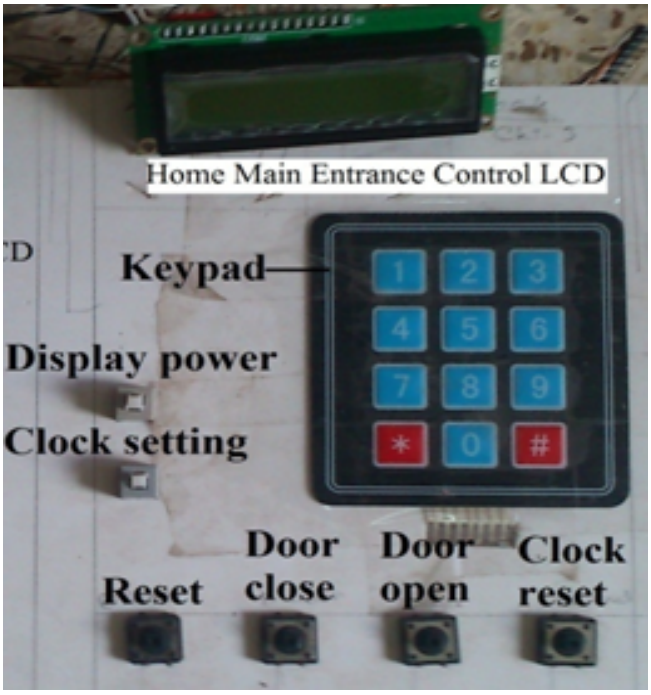

Figure 5.9: Keypad for main entrance security

5.2.2 Garage Framework

Garage framework consists of parking slot housing and display and mapping housing. Those are described below.

Parking Slot:

This part shows a demo of garage. A vehicle enters into the garage through the main gate. Here two pairs of infrared sensors are placed where one pair of infrared is placed in front of the gate and the other pair is placed after the gate bar. When vehicle tries to enter into the parking lot then these infrared sensors will sense the vehicle and the garage gate will open. Here, our garage system is designed for four vehicles. Vehicles are tracked by infrared sensors which are situated in every parking slot. Figure 5.10 shows the parking slot framework.

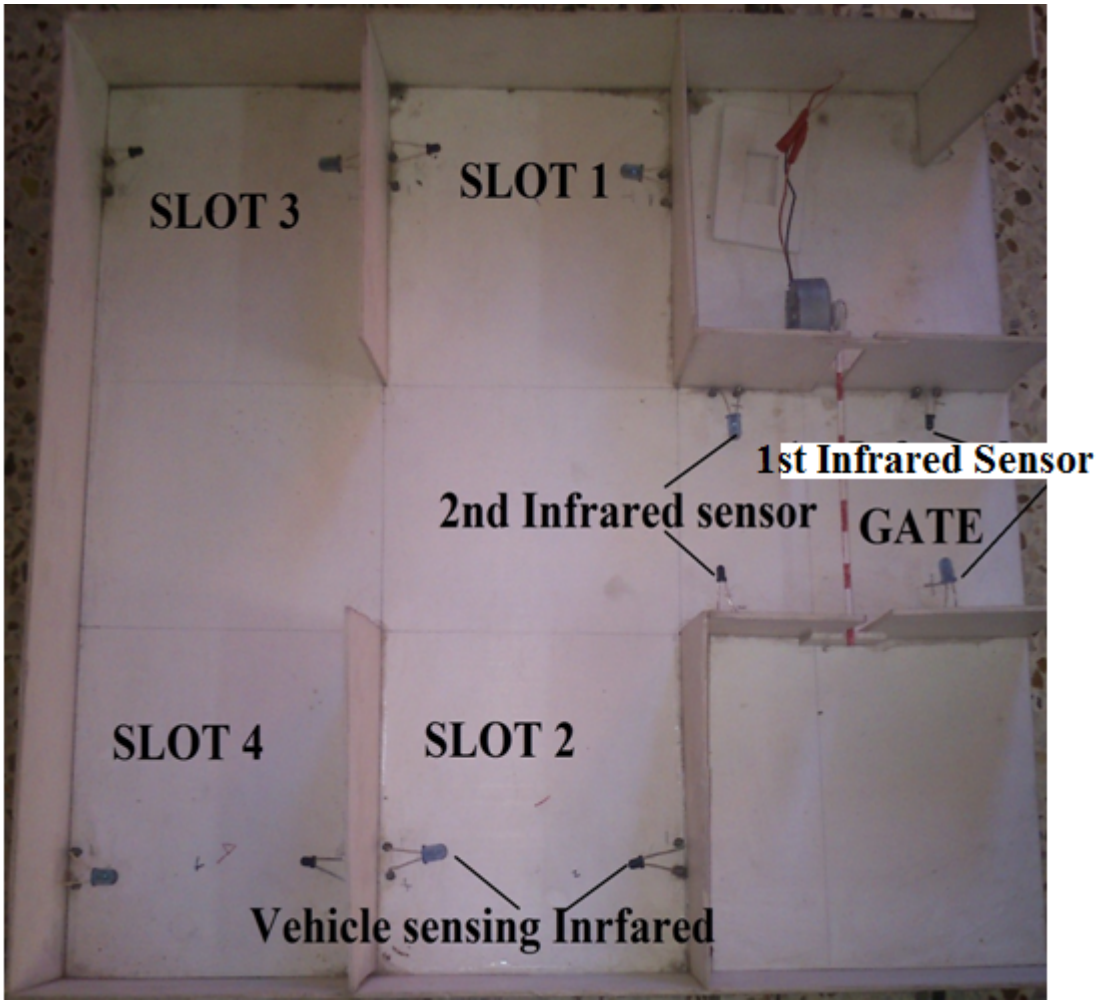

Figure 5.10: Garage Slot Framework

Display and Mapping:

When sensors of parking slot senses a vehicle then the system starts its counting. The counting result will be displayed in seven segment display which is shown in Figure 5.11. We also can get information about which parking slot is free and which are occupied from the LEDs map (Figure 5.12). The seven segment displays and the LED mapping are situated just in front of garage.

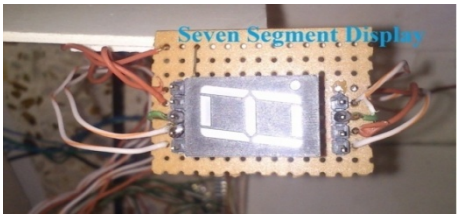

Figure 5.11: Seven Segment Display

There are five LEDs in the vehicle mapping. The first LED gives information whether the vehicle parking gate is opened or closed. If the parking gate is closed, the first LED will alight. The other four LEDs are for four parking slots. When the slots are free, the LEDs are off. When a vehicle comes to a free parking slot or the system books a slot for future use, the corresponding LED will turn on. Figure 5.12 shows the system built by us.

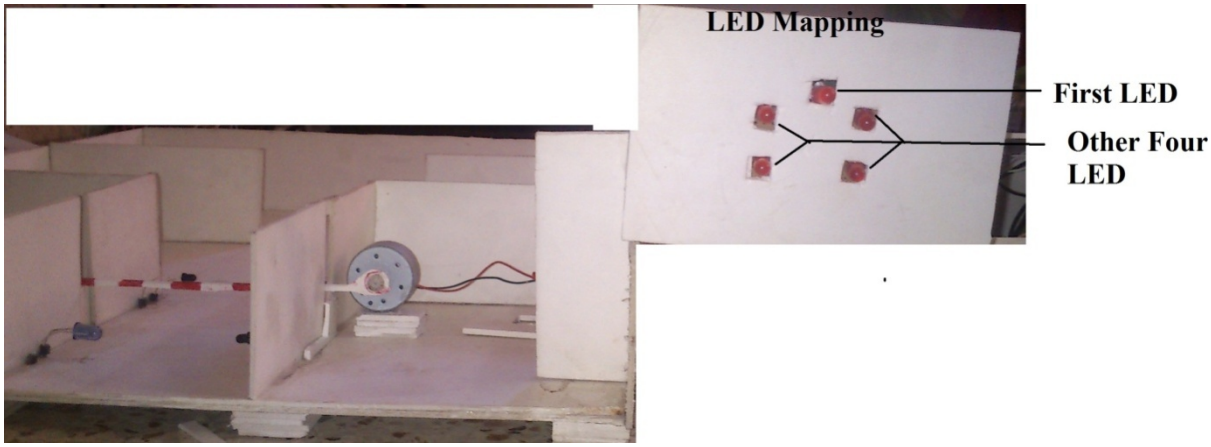

Figure 5.12: Vehicle Counting Display and Mapping by LED

Chapter 6
Conclusion

Microcontroller based smart home monitoring and control system is a complete solution for a home and modern garage. Security system of home entrance reduces the dependency on security guard. This system helps us to be safe from fire hazards. Home fixtures (i.e. lights, fans) are maintained and controlled by remote control system so physically handicapped persons can easily control home fixtures through remote control. S/he will not depend on other person for controlling those fixtures. If there are no persons in home, the lights and fans are turned off automatically. So we can reduce our power consumption. We monitor power consumption of the system through computer software. This feature will help us to estimate our electricity bill.

The garage system helps us to get all information of vehicle parking including the number of free parking slots, present number of vehicles and booked parking slots in a moment. We will get all these information from the garage mapping board and the computer software.

Computer software system makes it easier to control the whole the system by a single person. Thus lowering the maintenance cost.

Future development

The system is run by remote control and computer software. In future our plan is to develop internet based monitoring and controlling. Then we can control and monitor the system from anywhere in the world.

In our door security system, our plan is to the install an automatic phone call system to police station. Then if any unauthorized person tries to enter home by breaking the door, a voice call will be made to the police station.

References

[1] (2013, January) PIC18F4550 microcontroller datasheet. [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/39632e.pdf

[2] (2013, January) USB Transceiver. [Online]. Available: http://www.maximintegrated.com/products/interface/transceivers/usb-transceivers.cfm

[3] (2013, January) Universal serial bus. [Online]. Available: http://en.wikipedia.org/wiki/Universal_Serial_Bus

[4] (2013, January) Remote Control. [Online]. Available: http://www.edn.com/design/consumer/4313255/A-universal-algorithm-for-implementing-an-infrared-decoder

[5] (2013, February) SIRC protocol. [Online]. Available: http://picprojects.org.uk/projects/sirc/sonysirc.pdf

[6] (2013, February) Infrared sensor. [Online]. Available: http://www.wisegeek.org/what-is-an-infrared-sensor.html

[7] (2013, February) Temperature sensor. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm35.pdf

[8] (2013, February) DC motor. [Online]. Available: http://en.wikipedia.org/wiki/DC_motor

[9] (2013, February) DC motor driver pin description. [Online]. Available: http://www.engineersgarage.com/electronic-components/l293d-motor-driver-ic

[10] (2013, February) Datasheet of comparator LM 324. [Online]. Available: http://www.fairchildsemi.com/ds/LM/LM324.pdf

Appendix A

Security System Code:

```
#define rs LATA.LATA0
#define rw LATA.LATA1
#define en LATA.LATA2

//LCD Data pins
#define lcdport LATB

void lcd_ini();
int clk();
void lcdcmd(unsigned char);
void lcddata(unsigned char);
int s1=0,s2=0;
int m1=0,m2=0;
unsigned char txt[]="Ready";
unsigned char txt1[]="Player1";
unsigned char txt2[]="Player2";
unsigned char txt3[]="Time is up";
unsigned int i=0;

void lcd_ini()
{
    lcdcmd(0x38);          // Configure the LCD in 8-bit mode, 2 line and 5x7 font
    lcdcmd(0x0C);          // Display On and Cursor Off
    lcdcmd(0x01);          // Clear display screen
    lcdcmd(0x06);          // Increment cursor
    lcdcmd(0x80);          // Set cursor position to 1st line, 1st column
}



void lcdcmd(unsigned char cmdout)
{
    lcdport=cmdout;            //Send command to lcdport=PORTB
    rs=0;
    rw=0;
    en=1;
    Delay_ms(10);
    en=0;
}

void lcddata(unsigned char dataout)
{
    lcdport=dataout;      //Send data to lcdport=PORTB
    rs=1;
```

```
      rw=0;
      en=1;
      Delay_ms(10);
      en=0;
}


int clk()
{

 int j,k;
 int temp=0;

 i=0;
 j=0;
 k=0;


 for(i=0;i<=59;i++)
    { here:
          s1 = i/ 10;
          s2= i%10;

         lcdcmd(0xC3);
         lcddata(s1+48);
         lcddata(s2+48);
     for(temp=0;temp<10;temp++){
     if(Button(&PORTC, 2, 1, 0)){
     i=0;
     j=0;
     goto here;
     }
     if(Button(&PORTC, 6, 1, 0)){
     return 1;
     }
     Delay_ms(100);
     }
     if (i == 59)
      {
      i=0;
      j++;
      m1=j/10;
      m2=j%10;

         lcdcmd(0xC0);
         lcddata(m1+48);
         lcddata(m2+48);
         lcddata(':');
```

```
      }
      if(j>9){
      return 2;

      }


  }
}




void main() {
  int result=0;
  TRISA=0;              // Configure Port A as output port
       LATA=0;
       TRISB=0;              // Configure Port B as output port
       LATB=0;
       TRISC=0b01000111;
       lcd_ini();            // LCD initialization
       i=0;
       while(txt[i]!='\0'){
       lcddata(txt[i]);
       i++;
       }


     while(1){

          if(Button(&PORTC, 0, 1, 0)){
          lcdcmd(0x01);
          lcdcmd(0x80);
          i=0;
          while(txt1[i]!='\0'){
          lcddata(txt1[i]);
          i++;
          }
          m1=0;
          m2=0;
          s1=0;
          s2=0;
          lcdcmd(0xC0);
          lcddata(m1+48);
          lcddata(m2+48);
          lcddata(':');
          lcddata(s1+48);
          lcddata(s2+48);
          result=clk();
```

```
        }
        if(Button(&PORTC, 1, 1, 0)){
        lcdcmd(0x01);
        lcdcmd(0x80);
        i=0;
        while(txt2[i]!='\0'){
        lcddata(txt2[i]);
        i++;
        }

        m1=0;
        m2=0;
        s1=0;
        s2=0;
        lcdcmd(0xC0);
        lcddata(m1+48);
        lcddata(m2+48);
        lcddata(':');
        lcddata(s1+48);
        lcddata(s2+48);
        result=clk();
        }
        if(result==2){
        lcdcmd(0xC0);
        i=0;
        while(txt3[i]!='\0'){
        lcddata(txt3[i]);
        i++;
        }
        result=0;
        }
        if(result==1){
        lcdcmd(0x01);
        lcdcmd(0x80);
        i=0;
        while(txt[i]!='\0'){
        lcddata(txt[i]);
        i++;
        }
        result=0;
        }

    }
}
```

Garage GUI Code:

```
// GarageDlg.cpp : implementation file
//

#include "stdafx.h"
#include "Garage.h"
#include "GarageDlg.h"
#include "mcHID.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

// Dialog Data
	//{{AFX_DATA(CAboutDlg)
	enum { IDD = IDD_ABOUTBOX };
	//}}AFX_DATA

	// ClassWizard generated virtual function overrides
	//{{AFX_VIRTUAL(CAboutDlg)
	protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
	//}}AFX_VIRTUAL

// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
```

```
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CGarageDlg dialog

CGarageDlg::CGarageDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CGarageDlg::IDD, pParent)
	, function(0)
	, state(false)
{
	//{{AFX_DATA_INIT(CGarageDlg)
		// NOTE: the ClassWizard will add member initialization here
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CGarageDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CGarageDlg)
	// NOTE: the ClassWizard will add DDX and DDV calls here
	//}}AFX_DATA_MAP
	//  DDX_Control(pDX, IDC_MAP1, map1);
	//  DDX_Control(pDX, IDC_MAP2, map2);
	//  DDX_Control(pDX, IDC_MAP3, map3);
	//  DDX_Control(pDX, IDC_MAP4, map4);
	//  DDX_Control(pDX, IDC_SLOT1, slot1);
	//  DDX_Control(pDX, IDC_SLOT2, slot2);
	//  DDX_Control(pDX, IDC_SLOT3, slot3);
	//  DDX_Control(pDX, IDC_SLOT4, slot4);
	//  DDX_Control(pDX, IDC_NOR, nor);
	//  DDX_Control(pDX, IDC_OPE, ope);
	//  DDX_Control(pDX, IDC_CLO, clo);
	DDX_Control(pDX, IDC_SLOT1, slot1);
	DDX_Control(pDX, IDC_SLOT2, slot2);
	DDX_Control(pDX, IDC_SLOT3, slot3);
	DDX_Control(pDX, IDC_SLOT4, slot4);
	DDX_Control(pDX, IDC_NOR, nor);
```

```
	DDX_Control(pDX, IDC_OPE, ope);
	DDX_Control(pDX, IDC_CLO, clo);
	DDX_Control(pDX, IDC_MAP1, map1);
	DDX_Control(pDX, IDC_MAP2, map2);
	DDX_Control(pDX, IDC_MAP3, map3);
	DDX_Control(pDX, IDC_MAP4, map4);
	DDX_Control(pDX, IDC_0, num0);
	DDX_Control(pDX, IDC_1, num1);
	DDX_Control(pDX, IDC_2, num2);
	DDX_Control(pDX, IDC_3, num3);
	DDX_Control(pDX, IDC_4, num4);
	DDX_Control(pDX, IDC_5, num5);
	DDX_Control(pDX, IDC_6, num6);
	DDX_Control(pDX, IDC_7, num7);
	DDX_Control(pDX, IDC_8, num8);
	DDX_Control(pDX, IDC_9, num9);
	DDX_Control(pDX, IDC_FREE1, free1);
	DDX_Control(pDX, IDC_FREE2, free2);
	DDX_Control(pDX, IDC_FREE3, free3);
	DDX_Control(pDX, IDC_FREE4, free4);
	DDX_Control(pDX, IDC_UNBOOK1, ubook1);
	DDX_Control(pDX, IDC_UNBOOK2, ubook2);
	DDX_Control(pDX, IDC_UNBOOK3, ubook3);
	DDX_Control(pDX, IDC_UNBOOK4, ubook4);
	DDX_Control(pDX, IDC_UNOR1, unor1);
	DDX_Control(pDX, IDC_UNOR2, unor2);
	DDX_Control(pDX, IDC_UNOR3, unor3);
	DDX_Control(pDX, IDC_REAR_B2, rear_b2);
	DDX_Control(pDX, IDC_REAR_B4, rear_b4);
	//  DDX_Control(pDX, IDC_front_B3, front_b1);
	DDX_Control(pDX, IDC_front_B3, front_b3);
	DDX_Control(pDX, IDC_FONT_B, front_b1);
}

BEGIN_MESSAGE_MAP(CGarageDlg, CDialog)
	//{{AFX_MSG_MAP(CGarageDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_WM_CREATE()
	ON_MESSAGE(WM_HID_EVENT, OnHIDEvent)
	//}}AFX_MSG_MAP
	ON_BN_CLICKED(IDOK, &CGarageDlg::OnBnClickedOk)
	ON_BN_CLICKED(IDC_CNTPLUS, &CGarageDlg::OnBnClickedCntplus)
	ON_BN_CLICKED(IDC_CNTMINUS, &CGarageDlg::OnBnClickedCntminus)
	ON_BN_CLICKED(IDC_NORMAL, &CGarageDlg::OnBnClickedNormal)
	ON_BN_CLICKED(IDC_OPEN, &CGarageDlg::OnBnClickedOpen)
	ON_BN_CLICKED(IDC_CLOSE, &CGarageDlg::OnBnClickedClose)
	ON_BN_CLICKED(IDC_BOOK1, &CGarageDlg::OnBnClickedBook1)
```

```
	ON_BN_CLICKED(IDC_BOOK2, &CGarageDlg::OnBnClickedBook2)
	ON_BN_CLICKED(IDC_BOOK3, &CGarageDlg::OnBnClickedBook3)
	ON_BN_CLICKED(IDC_BOOK4, &CGarageDlg::OnBnClickedBook4)
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
// CGarageDlg message handlers

BOOL CGarageDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here
	num0.ShowWindow(SW_SHOW);
	num1.ShowWindow(SW_HIDE);
	num2.ShowWindow(SW_HIDE);
	num3.ShowWindow(SW_HIDE);
	num4.ShowWindow(SW_HIDE);
	num5.ShowWindow(SW_HIDE);
	num6.ShowWindow(SW_HIDE);
	num7.ShowWindow(SW_HIDE);
	num8.ShowWindow(SW_HIDE);
	num9.ShowWindow(SW_HIDE);

	map1.ShowWindow(SW_HIDE);
	map2.ShowWindow(SW_HIDE);
```

```
	map3.ShowWindow(SW_HIDE);
	map4.ShowWindow(SW_HIDE);

	free1.ShowWindow(SW_SHOW);
	free2.ShowWindow(SW_SHOW);
	free3.ShowWindow(SW_SHOW);
	free4.ShowWindow(SW_SHOW);



	slot1.ShowWindow(SW_HIDE);
	slot2.ShowWindow(SW_HIDE);
	slot3.ShowWindow(SW_HIDE);
	slot4.ShowWindow(SW_HIDE);

	ubook1.ShowWindow(SW_SHOW);
	ubook2.ShowWindow(SW_SHOW);
	ubook3.ShowWindow(SW_SHOW);
	ubook4.ShowWindow(SW_SHOW);


	nor.ShowWindow(SW_SHOW);
	ope.ShowWindow(SW_HIDE);
	clo.ShowWindow(SW_HIDE);

	unor1.ShowWindow(SW_HIDE);
	unor2.ShowWindow(SW_SHOW);
	unor3.ShowWindow(SW_SHOW);

	front_b1.ShowWindow(SW_HIDE);
	front_b3.ShowWindow(SW_HIDE);
	rear_b2.ShowWindow(SW_HIDE);
	rear_b4.ShowWindow(SW_HIDE);
	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CGarageDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}
```

```
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CGarageDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CGarageDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

/*
*********************************************************************************
* Name  : OnCreate                                                              *
* Notes : Create the dialog and connect to the HID DLL                          *
*********************************************************************************
*/
int CGarageDlg::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  if (CDialog::OnCreate(lpCreateStruct) == -1)
    return -1;

  // connect to the HID DLL
  CWnd *HostWnd = AfxGetMainWnd();
```

```
  if (LoadHID() == 0)
    Connect(HostWnd->m_hWnd);

  return 0;
}
/*
**************************************************************************
* Name  : OnHIDEvent                                    *
* Notes : Message handler for all HID events                    *
**************************************************************************
*/
LRESULT CGarageDlg::OnHIDEvent(WPARAM wParam, LPARAM lParam)
{
  CStatic* Label = (CStatic*) GetDlgItem(IDC_STATIC); // get dialog label
  char Text[0xFF];                    // text buffer
  UINT DevHandle;                     // HID device handle
  unsigned char BufferIN[100];                  // HID read buffer

  DevHandle = lParam;
  switch(wParam)
  {
            // a HID device has been plugged in...
                  case NOTIFY_PLUGGED:
                  {
                        // is it our HID device...
                        if (GetVendorID(DevHandle) == VENDOR_ID && GetProductID(DevHandle) ==
PRODUCT_ID)
                        {
                              GetProductName(DevHandle,Text,0xFF);
                              Label->SetWindowText(CString("USB Device '") +
                                          CString(Text) +
                                                      CString("' is PLUGGED
IN"));
                        }
                        state= TRUE;
                        return 1;
                  }

                  // a HID device has been removed...
                  case NOTIFY_UNPLUGGED:
                  {
                        // is it our HID device...
                        if (GetVendorID(DevHandle) == VENDOR_ID && GetProductID(DevHandle) ==
PRODUCT_ID)
                     Label->SetWindowText("USB Device has been UNPLUGGED");

                        state=FALSE;
                        return 1;
                  }
```

```
// a HID device has been pluggined in or removed - this event
// is fired after all PLUGGED and UNPLUGGED messages have
// been dispatched...
case NOTIFY_CHANGED:
{
	// we want our device to send read notification messages...
	DevHandle = GetHandle(VENDOR_ID, PRODUCT_ID);
	SetReadNotify(DevHandle, TRUE);
	return 1;
}

// a HID device has sent some data...
case NOTIFY_READ:
{
	// read the data...
	Read(DevHandle,BufferIN);

	// *** process data here **

	if(BufferIN[1]=='1' && BufferIN[5]=='1'){
		front_b1.ShowWindow(SW_SHOW);
		map1.ShowWindow(SW_HIDE);
		free1.ShowWindow(SW_HIDE);

	}else if(BufferIN[1]=='1' && BufferIN[5]=='0'){
		front_b1.ShowWindow(SW_HIDE);
		map1.ShowWindow(SW_SHOW);
		free1.ShowWindow(SW_HIDE);

	}else{
		front_b1.ShowWindow(SW_HIDE);
		map1.ShowWindow(SW_HIDE);
		free1.ShowWindow(SW_SHOW);
	}

	if(BufferIN[2]=='1' && BufferIN[6]=='1'){
		rear_b2.ShowWindow(SW_SHOW);
		map2.ShowWindow(SW_HIDE);
		free2.ShowWindow(SW_HIDE);

	}else if(BufferIN[2]=='1' && BufferIN[6]=='0'){
		rear_b2.ShowWindow(SW_HIDE);
		map2.ShowWindow(SW_SHOW);
		free2.ShowWindow(SW_HIDE);

	}else{
		rear_b2.ShowWindow(SW_HIDE);
		map2.ShowWindow(SW_HIDE);
```

```
		free2.ShowWindow(SW_SHOW);
	}

	if(BufferIN[3]=='1' && BufferIN[7]=='1'){
		front_b3.ShowWindow(SW_SHOW);
		map3.ShowWindow(SW_HIDE);
		free3.ShowWindow(SW_HIDE);

	}else if(BufferIN[3]=='1' && BufferIN[7]=='0'){
		front_b3.ShowWindow(SW_HIDE);
		map3.ShowWindow(SW_SHOW);
		free3.ShowWindow(SW_HIDE);

	}else{
		front_b3.ShowWindow(SW_HIDE);
		map3.ShowWindow(SW_HIDE);
		free3.ShowWindow(SW_SHOW);
	}


	if(BufferIN[4]=='1' && BufferIN[8]=='1'){
		rear_b4.ShowWindow(SW_SHOW);
		map4.ShowWindow(SW_HIDE);
		free4.ShowWindow(SW_HIDE);

	}else if(BufferIN[4]=='1' && BufferIN[8]=='0'){
		rear_b4.ShowWindow(SW_HIDE);
		map4.ShowWindow(SW_SHOW);
		free4.ShowWindow(SW_HIDE);

	}else{
		rear_b4.ShowWindow(SW_HIDE);
		map4.ShowWindow(SW_HIDE);
		free4.ShowWindow(SW_SHOW);
	}

	if(BufferIN[5]=='1'){
		slot1.ShowWindow(SW_SHOW);
		ubook1.ShowWindow(SW_HIDE);

	}else{
		slot1.ShowWindow(SW_HIDE);
		ubook1.ShowWindow(SW_SHOW);

	}
	if(BufferIN[6]=='1'){
		slot2.ShowWindow(SW_SHOW);
		ubook2.ShowWindow(SW_HIDE);
```

```
}else{
        slot2.ShowWindow(SW_HIDE);
        ubook2.ShowWindow(SW_SHOW);

}
if(BufferIN[7]=='1'){
        slot3.ShowWindow(SW_SHOW);
        ubook3.ShowWindow(SW_HIDE);

}else{
        slot3.ShowWindow(SW_HIDE);
        ubook3.ShowWindow(SW_SHOW);

}
if(BufferIN[8]=='1'){
        slot4.ShowWindow(SW_SHOW);
        ubook4.ShowWindow(SW_HIDE);

}else{
        slot4.ShowWindow(SW_HIDE);
        ubook4.ShowWindow(SW_SHOW);

}
if(BufferIN[10]=='0'){
        nor.ShowWindow(SW_SHOW);
        unor1.ShowWindow(SW_HIDE);

}else{
        nor.ShowWindow(SW_HIDE);
        unor1.ShowWindow(SW_SHOW);

}
if(BufferIN[10]=='1'){
        ope.ShowWindow(SW_SHOW);
        unor2.ShowWindow(SW_HIDE);

}else{
        ope.ShowWindow(SW_HIDE);
        unor2.ShowWindow(SW_SHOW);

}
if(BufferIN[10]=='2'){
        clo.ShowWindow(SW_SHOW);
        unor3.ShowWindow(SW_HIDE);

}else{
        clo.ShowWindow(SW_HIDE);
        unor3.ShowWindow(SW_SHOW);
```

```
}


if(BufferIN[9]=='0'){

   num0.ShowWindow(SW_SHOW);
        num1.ShowWindow(SW_HIDE);
        num2.ShowWindow(SW_HIDE);
        num3.ShowWindow(SW_HIDE);
        num4.ShowWindow(SW_HIDE);
        num5.ShowWindow(SW_HIDE);
        num6.ShowWindow(SW_HIDE);
        num7.ShowWindow(SW_HIDE);
        num8.ShowWindow(SW_HIDE);
        num9.ShowWindow(SW_HIDE);
}else if(BufferIN[9]=='1'){

   num0.ShowWindow(SW_HIDE);
        num1.ShowWindow(SW_SHOW);
        num2.ShowWindow(SW_HIDE);
        num3.ShowWindow(SW_HIDE);
        num4.ShowWindow(SW_HIDE);
        num5.ShowWindow(SW_HIDE);
        num6.ShowWindow(SW_HIDE);
        num7.ShowWindow(SW_HIDE);
        num8.ShowWindow(SW_HIDE);
        num9.ShowWindow(SW_HIDE);
}else if(BufferIN[9]=='2'){

   num0.ShowWindow(SW_HIDE);
        num1.ShowWindow(SW_HIDE);
        num2.ShowWindow(SW_SHOW);
        num3.ShowWindow(SW_HIDE);
        num4.ShowWindow(SW_HIDE);
        num5.ShowWindow(SW_HIDE);
        num6.ShowWindow(SW_HIDE);
        num7.ShowWindow(SW_HIDE);
        num8.ShowWindow(SW_HIDE);
        num9.ShowWindow(SW_HIDE);
}else if(BufferIN[9]=='3'){

   num0.ShowWindow(SW_HIDE);
        num1.ShowWindow(SW_HIDE);
        num2.ShowWindow(SW_HIDE);
        num3.ShowWindow(SW_SHOW);
        num4.ShowWindow(SW_HIDE);
        num5.ShowWindow(SW_HIDE);
        num6.ShowWindow(SW_HIDE);
        num7.ShowWindow(SW_HIDE);
```

```
            num8.ShowWindow(SW_HIDE);
            num9.ShowWindow(SW_HIDE);
        }else if(BufferIN[9]=='4'){

          num0.ShowWindow(SW_HIDE);
            num1.ShowWindow(SW_HIDE);
            num2.ShowWindow(SW_HIDE);
            num3.ShowWindow(SW_HIDE);
            num4.ShowWindow(SW_SHOW);
            num5.ShowWindow(SW_HIDE);
            num6.ShowWindow(SW_HIDE);
            num7.ShowWindow(SW_HIDE);
            num8.ShowWindow(SW_HIDE);
            num9.ShowWindow(SW_HIDE);
        }else if(BufferIN[9]=='5'){

          num0.ShowWindow(SW_HIDE);
            num1.ShowWindow(SW_HIDE);
            num2.ShowWindow(SW_HIDE);
            num3.ShowWindow(SW_HIDE);
            num4.ShowWindow(SW_HIDE);
            num5.ShowWindow(SW_SHOW);
            num6.ShowWindow(SW_HIDE);
            num7.ShowWindow(SW_HIDE);
            num8.ShowWindow(SW_HIDE);
            num9.ShowWindow(SW_HIDE);
        }else if(BufferIN[9]=='6'){

          num0.ShowWindow(SW_HIDE);
            num1.ShowWindow(SW_HIDE);
            num2.ShowWindow(SW_HIDE);
            num3.ShowWindow(SW_HIDE);
            num4.ShowWindow(SW_HIDE);
            num5.ShowWindow(SW_HIDE);
            num6.ShowWindow(SW_SHOW);
            num7.ShowWindow(SW_HIDE);
            num8.ShowWindow(SW_HIDE);
            num9.ShowWindow(SW_HIDE);
        }else if(BufferIN[9]=='7'){

          num0.ShowWindow(SW_HIDE);
            num1.ShowWindow(SW_HIDE);
            num2.ShowWindow(SW_HIDE);
            num3.ShowWindow(SW_HIDE);
            num4.ShowWindow(SW_HIDE);
            num5.ShowWindow(SW_HIDE);
            num6.ShowWindow(SW_HIDE);
            num7.ShowWindow(SW_SHOW);
            num8.ShowWindow(SW_HIDE);
```

```
                                num9.ShowWindow(SW_HIDE);
                        }else if(BufferIN[9]=='8'){

                           num0.ShowWindow(SW_HIDE);
                                num1.ShowWindow(SW_HIDE);
                                num2.ShowWindow(SW_HIDE);
                                num3.ShowWindow(SW_HIDE);
                                num4.ShowWindow(SW_HIDE);
                                num5.ShowWindow(SW_HIDE);
                                num6.ShowWindow(SW_HIDE);
                                num7.ShowWindow(SW_HIDE);
                                num8.ShowWindow(SW_SHOW);
                                num9.ShowWindow(SW_HIDE);
                        }else if(BufferIN[9]=='9'){

                           num0.ShowWindow(SW_HIDE);
                                num1.ShowWindow(SW_HIDE);
                                num2.ShowWindow(SW_HIDE);
                                num3.ShowWindow(SW_HIDE);
                                num4.ShowWindow(SW_HIDE);
                                num5.ShowWindow(SW_HIDE);
                                num6.ShowWindow(SW_HIDE);
                                num7.ShowWindow(SW_HIDE);
                                num8.ShowWindow(SW_HIDE);
                                num9.ShowWindow(SW_SHOW);
                        }


                        // first element BufferIn[0] is the report ID, the
                        // rest of the buffer contains the data sent from
                        // the HID device
                        return 1;
                }

  }
  Read(DevHandle,BufferIN);
  return 0;
}
/*

// To send some data to the HID device, use something like..
void CGarageDlg::SendSomeData()
{
  // output buffer...
  HIDBufferOut BufferOut;

  // first element is the report ID
  BufferOut[0] = 0;
```

```
  // rest of the buffer is for data...
  for (int i = 1; i < BUFFER_OUT_SIZE; i++)
    BufferOut[i] = i;

  // now send the data...
  WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
}

*/



        void CGarageDlg::OnBnClickedOk()
{
        // TODO: Add your control notification handler code here
        CDialog::OnOK();
}


        void CGarageDlg::OnBnClickedCntplus()
        {
                // TODO: Add your control notification handler code here

                HIDBufferOut BufferOut;
                BufferOut[0] = 0;
                if (state){
                BufferOut[6] = '1';
                WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
                UpdateData(FALSE);
                BufferOut[6] = '0';
                WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
                UpdateData(FALSE);
                }

        }


        void CGarageDlg::OnBnClickedCntminus()
        {
                // TODO: Add your control notification handler code here
                HIDBufferOut BufferOut;
                BufferOut[0] = 0;
                if (state){
                BufferOut[7] = '1';
                WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
                UpdateData(FALSE);
                BufferOut[7] = '0';
                WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
                UpdateData(FALSE);
```

```
		}


	}


	void CGarageDlg::OnBnClickedNormal()
	{
		// TODO: Add your control notification handler code here
		HIDBufferOut BufferOut;
		BufferOut[0] = 0;
		if (state){
		BufferOut[5] = '0';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		}

	}


	void CGarageDlg::OnBnClickedOpen()
	{
		// TODO: Add your control notification handler code here
		HIDBufferOut BufferOut;
		BufferOut[0] = 0;
		if (state){
		BufferOut[5] = '1';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		}

	}


	void CGarageDlg::OnBnClickedClose()
	{
		// TODO: Add your control notification handler code here
		HIDBufferOut BufferOut;
		BufferOut[0] = 0;
		if (state){
		BufferOut[5] = '2';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		}
	}


	void CGarageDlg::OnBnClickedBook1()
	{
```

```
		// TODO: Add your control notification handler code here
		HIDBufferOut BufferOut;
		BufferOut[0] = 0;
		if (state){
		BufferOut[1] = '1';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		BufferOut[1] = '0';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		}

}


void CGarageDlg::OnBnClickedBook2()
{
		// TODO: Add your control notification handler code here
		HIDBufferOut BufferOut;
		BufferOut[0] = 0;
		if (state){
		BufferOut[2] = '1';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		BufferOut[2] = '0';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		}

}


void CGarageDlg::OnBnClickedBook3()
{
		// TODO: Add your control notification handler code here
		HIDBufferOut BufferOut;
		BufferOut[0] = 0;
		if (state){
		BufferOut[3] = '1';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		BufferOut[3] = '0';
		WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
		UpdateData(FALSE);
		}

}
```

```
void CGarageDlg::OnBnClickedBook4()
{
	// TODO: Add your control notification handler code here
	HIDBufferOut BufferOut;
	BufferOut[0] = 0;
	if (state){
	BufferOut[4] = '1';
	WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
	UpdateData(FALSE);
	BufferOut[4] = '0';
	WriteEx(VENDOR_ID, PRODUCT_ID, BufferOut);
	UpdateData(FALSE);
	}

}
```