MUSEUM SECURITY SYSTEM

by

TASEEN MUHTADI

NURUL AMIN

MUSRAT TABASSUM

A thesis submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Science in Electrical &
Communication Engineering

BRAC University

December 2012

Declaration

We do hereby declare that the thesis titled "Museum Security System" is submitted to the Department of Electrical and Electronics Engineering of BRAC University in partial fulfillment of the Bachelor of Science in Electrical and Electronics Engineering. This is our original work and was not submitted elsewhere for the award of any other degree or any other publication.

Date: 15.12.2012

Supervisor

_____

SYED SHAKIB SARWAR

_____

TASEEN MUHTADI
07210040

_____

NURUL AMIN
07310058

_____

MUSRAT TABBASSUM
07210021

Acknowledgements

The authors wish to express sincere appreciation to Lecturer Syed Shakib Sarwar for his guidance and help in every step of this project. His familiarity with relevant fields and assistance with both software and hardware sides of the project allowed us to complete the project in such a short timeframe. In addition, special thanks to Dr. Amithabha Chakrabarty whose assistance in the early stages made this project possible. Thanks also to the Laboratory Technical Officers for their valuable inputs.

Abstract

A comprehensive security system incorporating motion sensors and laser sensors activated and deactivated by a password. The security system is designed to protect a valuable object. Access to the room can only be gained by entering the correct password. Upon unauthorized entry into the room or coming into close proximity of the object or removal of the object in question from its place would sound the alarm.

Table of Contents

Introduction

Motivation

Security systems in Bangladesh have always been manual. Old fashioned lock and keys are the main security devices used. Close circuit cameras are being used nowadays. But they have the disadvantage of being expensive. Another problem that comes with them is that people have to seat in front of monitors constantly watching the feed. If something were done when no one was watching or if the watcher failed to notice the incident then the theft of a valuable object would be undetectable. To add another layer of security to particularly important objects would increase their safety immensely and reduce lapses of security due to human error. Our system is fully automated once activated. It does not require anyone to continuously stand guard. If an intrusion or theft occurs only then will the authorities be alerted. It adds security without increasing the workload of the people tasked with security. It also scalable, the number of sensors can be increased and any number of the system can be operational in a given facility. Since our system is room and object specific, authorities will instantly know where to focus their attention when an alarm goes off. Our system being relatively inexpensive, it will add extra security without a huge investment in new equipment.

Summary of following chapters

In the following chapters we will be describing the overall system design, circuit design of the individual sensors, which includes a motion detector and multiple laser detectors; other hardware used and how they work. We will also discuss how they function as a whole and how everything is brought together using a microcontroller. The software side will also be discussed, along with the features and functions of our system.

Hardware used

Microcontroller

First, a little background on microcontrollers since our system is based on them. A **microcontroller** (sometimes abbreviated **µC**, **uC** or **MCU**) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications.

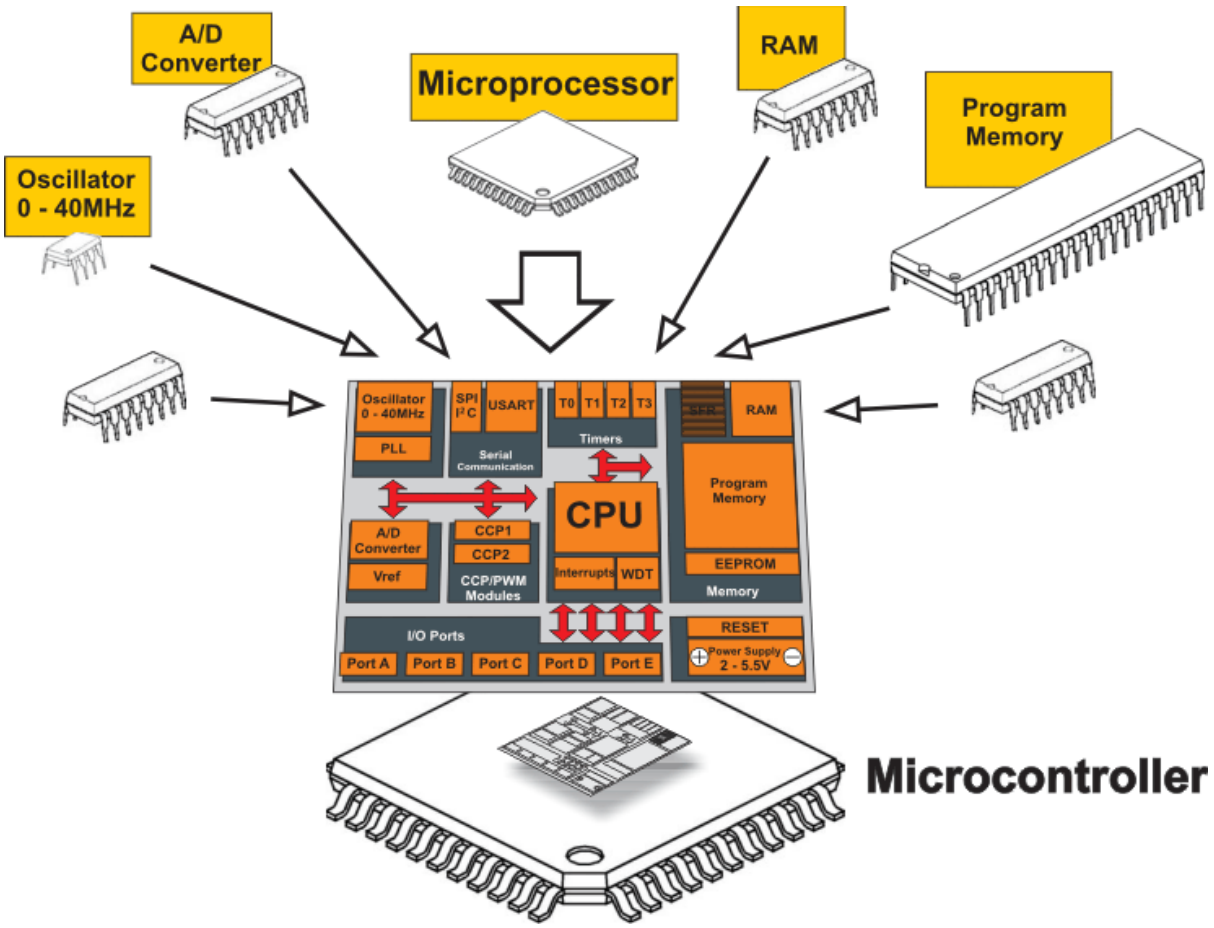

Figure 1

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and

input/output devices, microcontrollers make it economical to digitally control even more devices and processes. Mixed signal microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

The microcontrollers we used were PIC16F877A. It's a fairly large 40-pin microcontroller with 5 ports and 8Kb of flash program memory and 15 interrupts. It's a very powerful chip meant for complicated tasks. We chose this because we needed a minimum of 3 full ports. We also wanted the freedom to experiment with our software and hardware setup without having to worry about the limitations of the chip. This also gave us the option to increase the number and kind of sensors we used in our system.

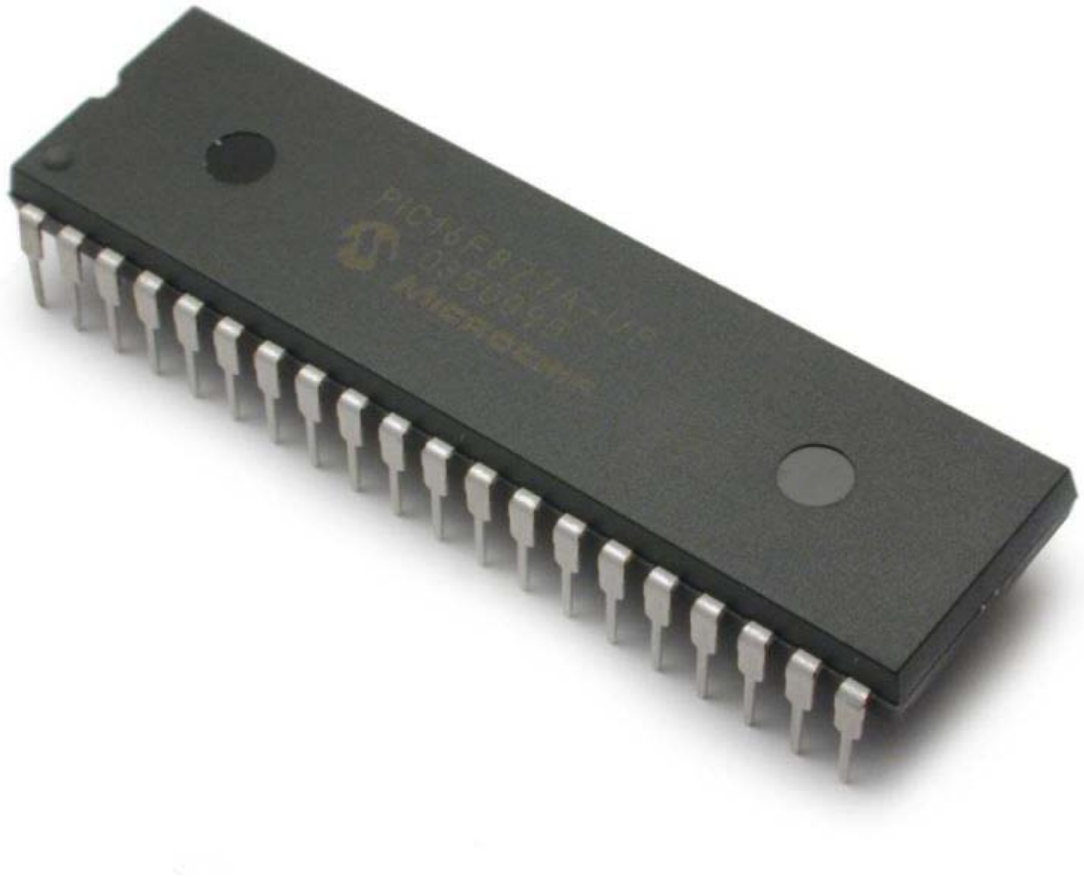

Figure 2

Motion sensor

Motion sensors are sensors that can sense movement. If a light has a motion sensor, it turns on when it senses movement.

There are two types of motion sensors currently commercially available, the active sensors and passive sensors. A motion sensor is classified as being active only when it emits some kind of energy into the surrounding medium to make an accurate reading, whether it is infrared light, microwave radiation or sound waves. Passive motion sensors do not emit energy, but can identify possible burglars by reading relative changes in the energy in the surrounding medium.

Figure 3

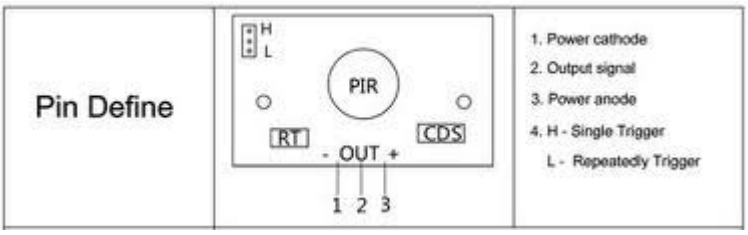

Figure 4

There are many different ways to create motion energy. Motion is detected when an infrared emitting source with one temperature, such as a human body, passes in front of a source with another temperature, such as a wall. It is common for stores to have a beam of light crossing the room near the door, and a photo sensor on the other

side of the room. When a customer breaks the beam, the photo sensor detects the change in the amount of light and rings a bell. Many grocery stores have automatic door openers that use a very simple form of radar to detect when someone passes near the door. The box above the door sends out a burst of microwave radio energy and waits for the reflected energy to bounce back. When a person moves into the field of microwave energy, it changes the amount of reflected energy or the time it takes for the reflection to arrive, and the box opens the door. Since these devices use radar, they often set off radar detectors. The same thing can be done with ultrasonic sound waves, bouncing them off a target and waiting for the echo.

All of these are active sensors. They inject energy (light, microwaves or sound) into the environment in order to detect a change of some sort.

The motion sensing feature on most lights is a passive system that detects infrared energy. These sensors are therefore known as PIR detectors sensors. Often, PIR technology will be paired with another model to maximize accuracy and reduce energy usage. In order to make a sensor that can detect a human being, you need to make the sensor sensitive to the temperature of a human body. Humans, having a skin temperature of about 93 degrees F, radiate infrared energy with a wavelength between 9 and 10 micrometers. Therefore, the sensors are typically sensitive in the range of 8 to 12 micrometers.

The devices themselves are simple electronic components not unlike a photo sensor. The infrared light bumps electrons off a substrate, and these electrons can be detected and amplified into a signal.

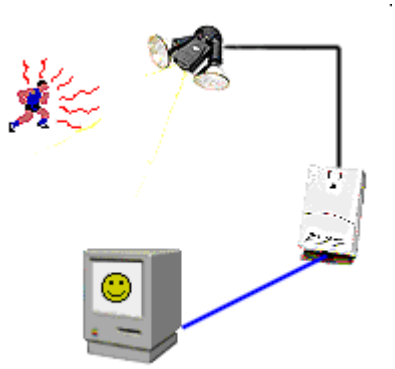

Figure 5

We have probably noticed that our detector is sensitive to motion, but not to a person who is standing still. That's because the electronics package attached to the sensor is looking for a fairly rapid change in the amount of infrared energy it is seeing. When a person walks by, the amount of infrared energy in the field of view changes rapidly and is easily detected. We do not want the sensor detecting slower changes, like the sidewalk cooling off at night.

Our motion sensor has a wide field of view because of the lens covering the sensor. Infrared energy is a form of light, so we can focus and bend it with plastic lenses. But it's not like there is a 2-D array of sensors in there. There is a single or sometimes two sensors inside looking for changes in infrared energy.

If we have a burglar alarm with motion sensors, we may have noticed that the motion sensors cannot "see" us when we are outside looking through a window. That is because glass is not very transparent to infrared energy. This, by the way, is the basis of a greenhouse. Light passes through the glass into the greenhouse and heats things up inside the greenhouse. The glass is then opaque to the infrared energy these heated things are emitting, so the heat is trapped inside the greenhouse. It makes sense that a motion detector sensitive to infrared energy cannot see through glass windows.

Light Dependent Resistor

An LDR is a component that has a resistance that changes with the intensity of light that falls upon it. They have a resistance that falls with an increase in the light intensity falling upon the device.

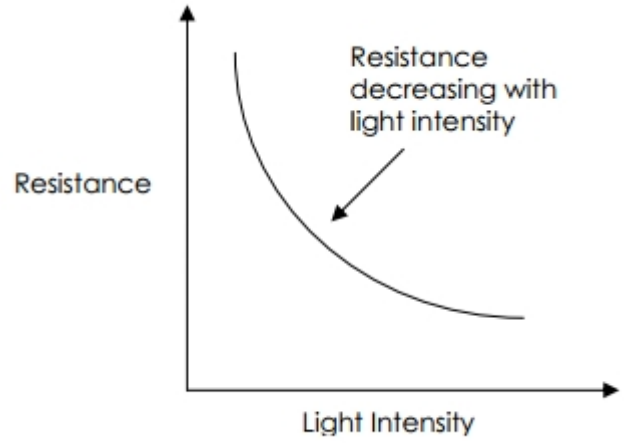

Figure 6 The relationship between the intensity of light and resistance of the LDR

The resistance of an LDR may typically have the following resistances.

Daylight = 5000 ohms
Dark = 20000000 ohms

We can therefore see that there is a large variation between these figures. If we plotted this variation on a graph we would get something similar to that shown by the graph to the right.

We have used the LDR coupled with a laser as a way of detecting unauthorized entry in the room. It can also be used to give a proximity warning in case someone comes too close to the object being guarded. We have two such circuits in our setup.

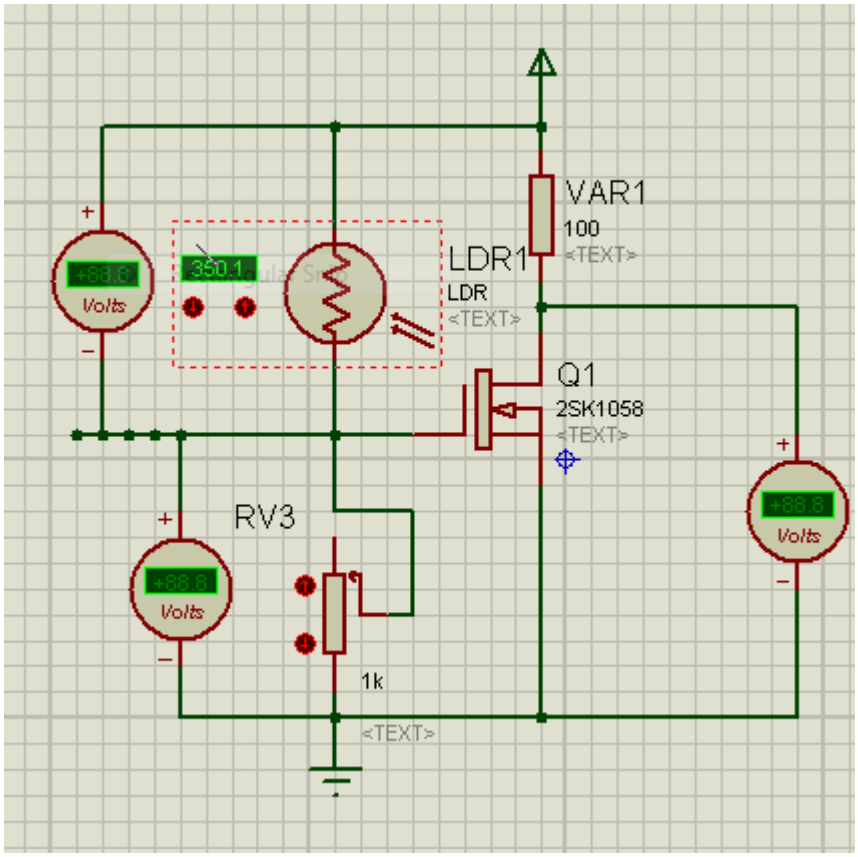

Figure 7 LDR Circuit

We used the above circuit in our project. We used IRFN150N MOSFET, a resistor, a variable resistor and one LDR to implement this circuit.

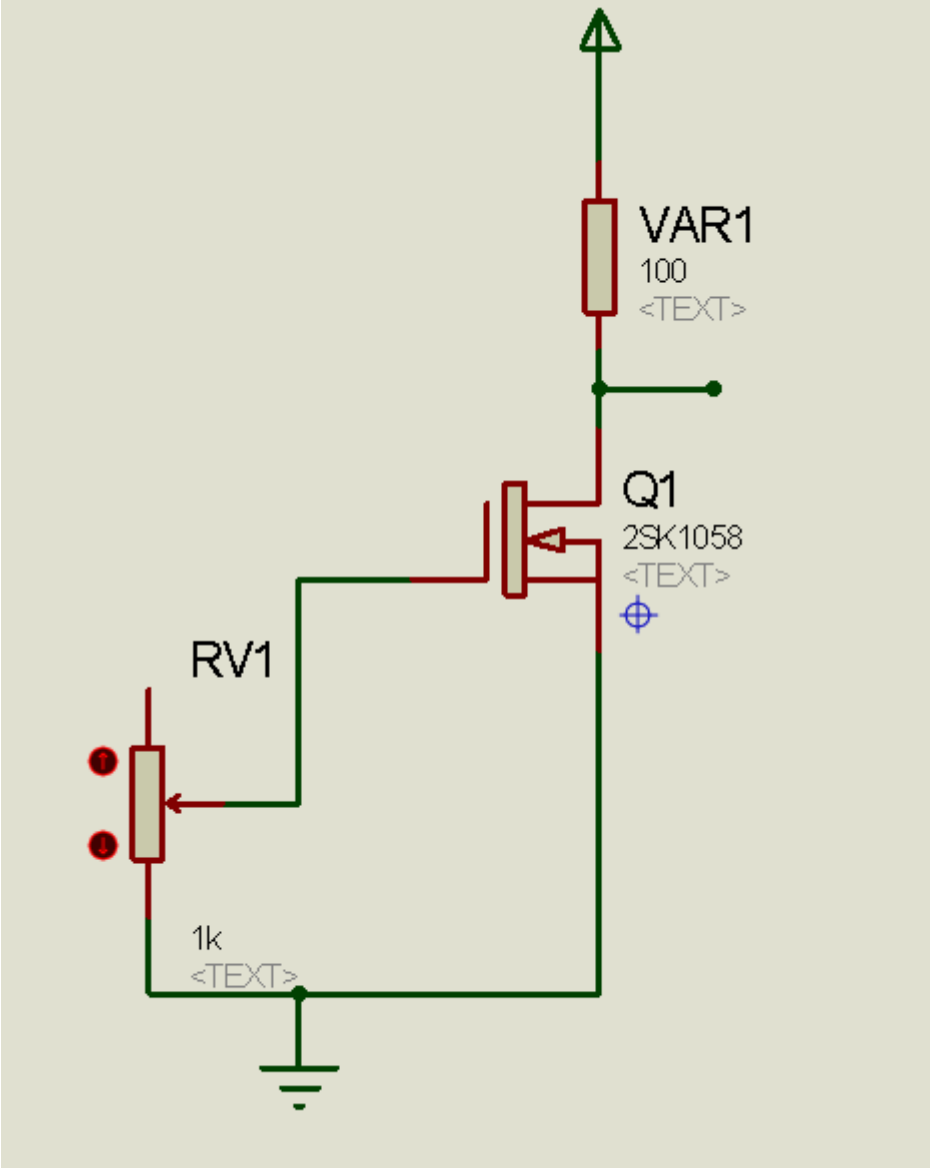

Figure 8 MOSFET Switch Circuit

This part of the circuit, with the variable resistor connected to the gate of the MOSFET acts as a switch. When the voltage at the gate reaches the threshold voltage of the MOSFET the Source and Drain of the MOSFET are shorted and output voltage drops to 0V. Normally the output voltage is 5V. This output is the input for the Microcontroller. The MOSFET switch circuit is also used with the motion sensor. We used these particular sensors since they are commercially available, fairly reliable and cost effective.

Keypad

Figure 9 4x4 Matrix Keypad

The keypad we used was a 4x4 matrix keypad. It has 16 buttons in total: 0-9, A-D and # and *. The keypad is the only user input device of our system; it is only used to enter the password that allows access to the room or object. We do not use 4 buttons; A, B, D and #. Our software is programmed to keep the keypad inactive under normal circumstances. Until the * button is pressed any other key press on the keypad is ignored. Once the * button is pressed the microcontroller will start accepting new key presses. The C button is used as a 'refresh' button of sorts. Pressing it will restart password entry process given the 4th digit is not yet entered. The schematic diagram of the keypad is given below.

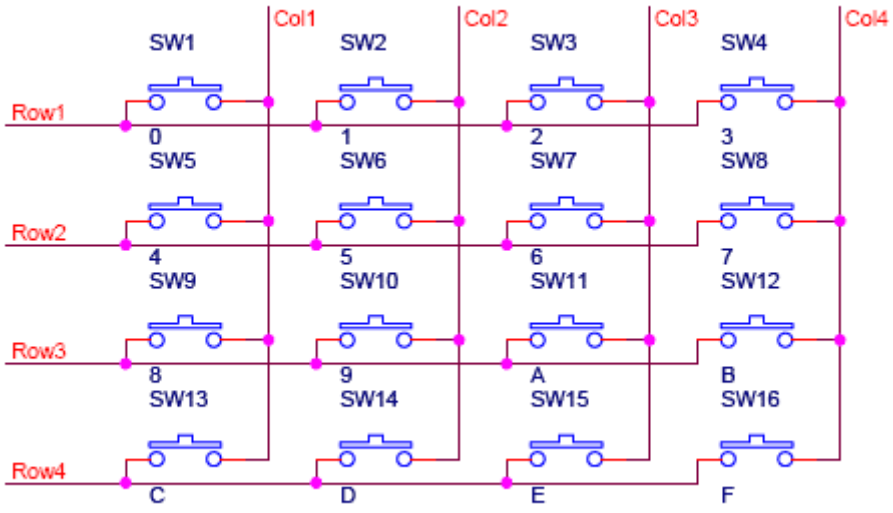

Figure 10 Schematic Diagram of 4x4 Matrix Keypad

It is interfaced with the microcontroller through a procedure called scanning. We make the columns as i/p and we drive the rows making them o/p.

In order to detect which key is pressed from the matrix, we make row lines low one by one and read the columns. Let's say we first make Row1 low, then read the columns. If any of the key in row1 is pressed will make the corresponding column as low i.e if second key is pressed in Row1, then column2 will give low. So we come to know that key 2 of Row1 is pressed. This is how scanning is done.

So to scan the keypad completely, we need to make rows low one by one and read the columns. If any of the buttons is pressed in a row, it will take the corresponding column to a low state which tells us that a key is pressed in that row. If button 1 of a row is pressed then Column 1 will become low, if button 2 then column2 and so on...

Circuit Design

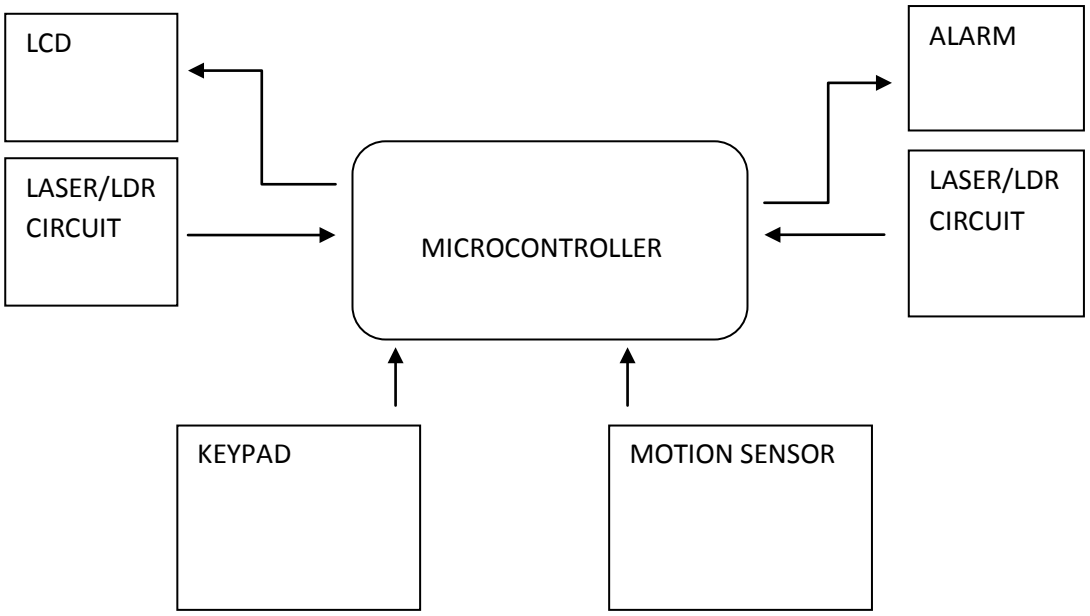

Figure 11 Block Diagram of circuit

The block diagram of our system is given above. This is the basic circuit setup we have used for demonstration purposes. This circuit has 2 Laser/LDR circuits, 1 Motion sensor and 1 Pressure sensor. This allows us to demonstrate the basic functionalities of the system. Inputs from all the sensors come to the microcontroller and the only output from the microcontroller goes to the alarm. Whenever any of the sensors detect something, the alarm goes off. The alarm is also set to go off if the wrong password is entered three times. This prevents anyone from stumbling across the correct password by repeatedly entering the various combinations. Once the wrong password has been entered three times the system stops taking any inputs from the keypad. The system will need to be reset for us to start taking input from the keypad again. An LCD is used to display messages regarding the current status of the system.

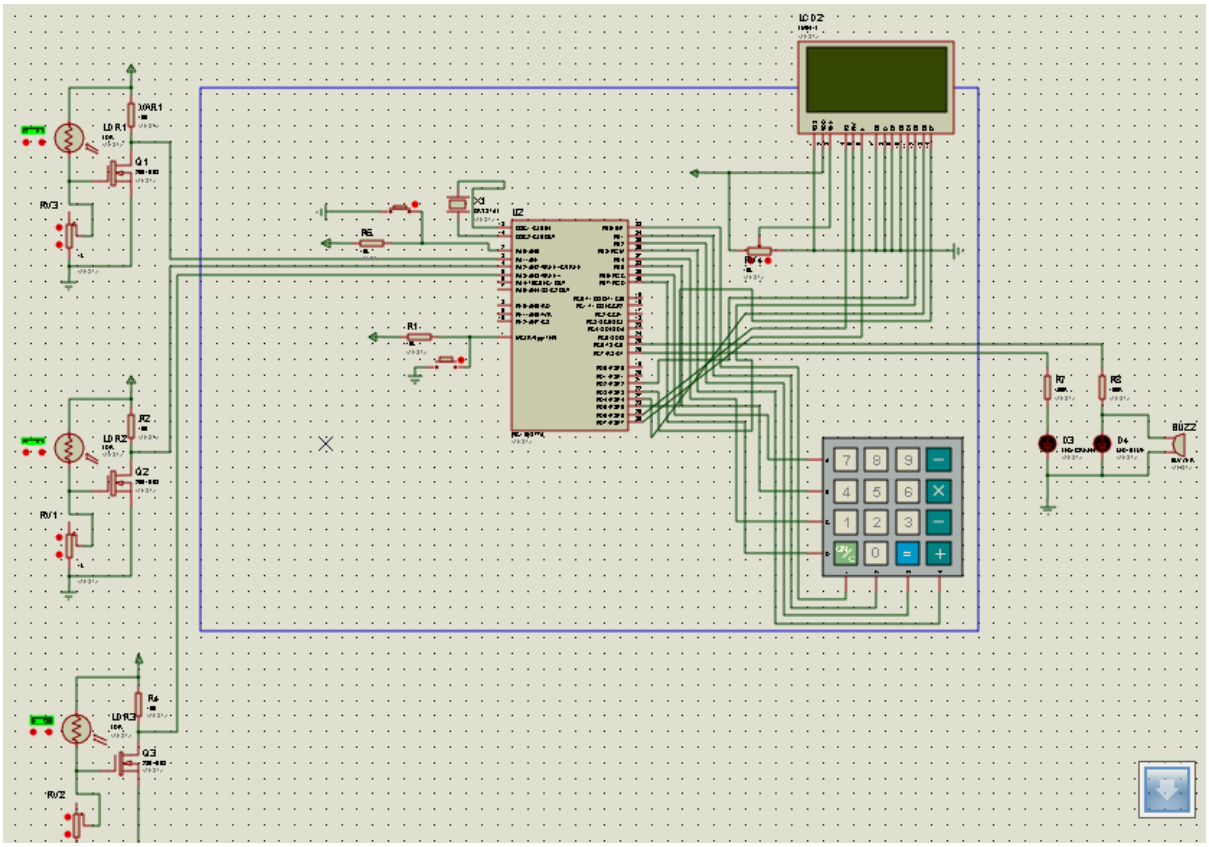

Figure 12 Schematic of Circuit

The schematic design of our system is given above. This is our full security system with laser/LDRs, motion sensors, pressure sensors and alarm.

Software Programming

The microcontroller has to be programed to function as we need it to. All the outputs from the all the sensors go to the microcontroller as its inputs. The microcontroller is then programed to respond to those inputs as necessary. This considerably simplifies our system in terms of hardware setup. Everything is connected to the microcontroller and we just have program it according to our needs. This also gives us the freedom to change the coding in case something does not work or we feel that we need to do things differently. The main advantage that we gain from the microcontroller is the ability to operate the sensors with a password.

The most important part the microcontroller plays is taking inputs from the keypad and checking it against a preprogrammed password. Only when the password is correct will it allow someone to enter the room and remove the object its guarding without set off the alarm. The software programmed in the microcontroller is centred around this function. An additional feature we have is that if the wrong password is entered 3 times in a row, the alarm will go off and the system will stop taking any further inputs.

The password is the "key" of our security system. The software written for our system mainly revolves around password operating all our sensors. The password itself is preprogrammed into the microcontroller. This preprogrammed password is checked against entries from the keypad. (how the keypad functions is covered in a previous section.) If the entered password is correct then all the sensors and the alarm are deactivated. If the wrong password is entered, the user will get 2 more attempts to enter the right password. If the user does not enter the correct password during the next 2 attempts the alarm will go off and the system will stop taking any further inputs from the keypad. The system will need to be reset with MCLR button on the microcontroller to resume normal operations again.

The reprogrammable microcontroller gives us enormous flexibility with the system design and functionality. We can add more sensors; adding more pressure sensors, for example, will allow us to guard more than one valuable object.

The software is written in the compiler "mikroC PRO for PIC." This compiler then creates a ".hex" file from the code. This ".hex" is then written or "burned" on to the microcontroller by a "PIC PROGRAMMER" or "burner." The program once written can be erased if the need arises. There is virtually no limit to how many times a given microcontroller can be reprogrammed due its use of flash memory. This is hugely useful during circuit debugging.

Results

When we were done with simulation, we began setting up our circuit in hardware. However we faced some major problems when we attempted to do so. Some of our equipment malfunctioned and due to time constraints we could not implements some of the programmed functions in hardware.

The first difficulty we faced was the malfunction of our LCD. Without a functioning display we could not implement the keypad functions. Thus we could not realize the functions that relied on the keypad and display; namely the password entry function.

We were however able to implement the various sensors and couple them with the microcontroller. We used a buzzer and LEDS as our output devices. The program was changed to light a LED whenever any of the sensors detected anything. The main difference between this setup and the one we originally designed was the lack of a password. Without password control over the sensors and alarm, they would go off anytime one of sensors detected an intrusion but we could not stop the alarm unless we reset the system.

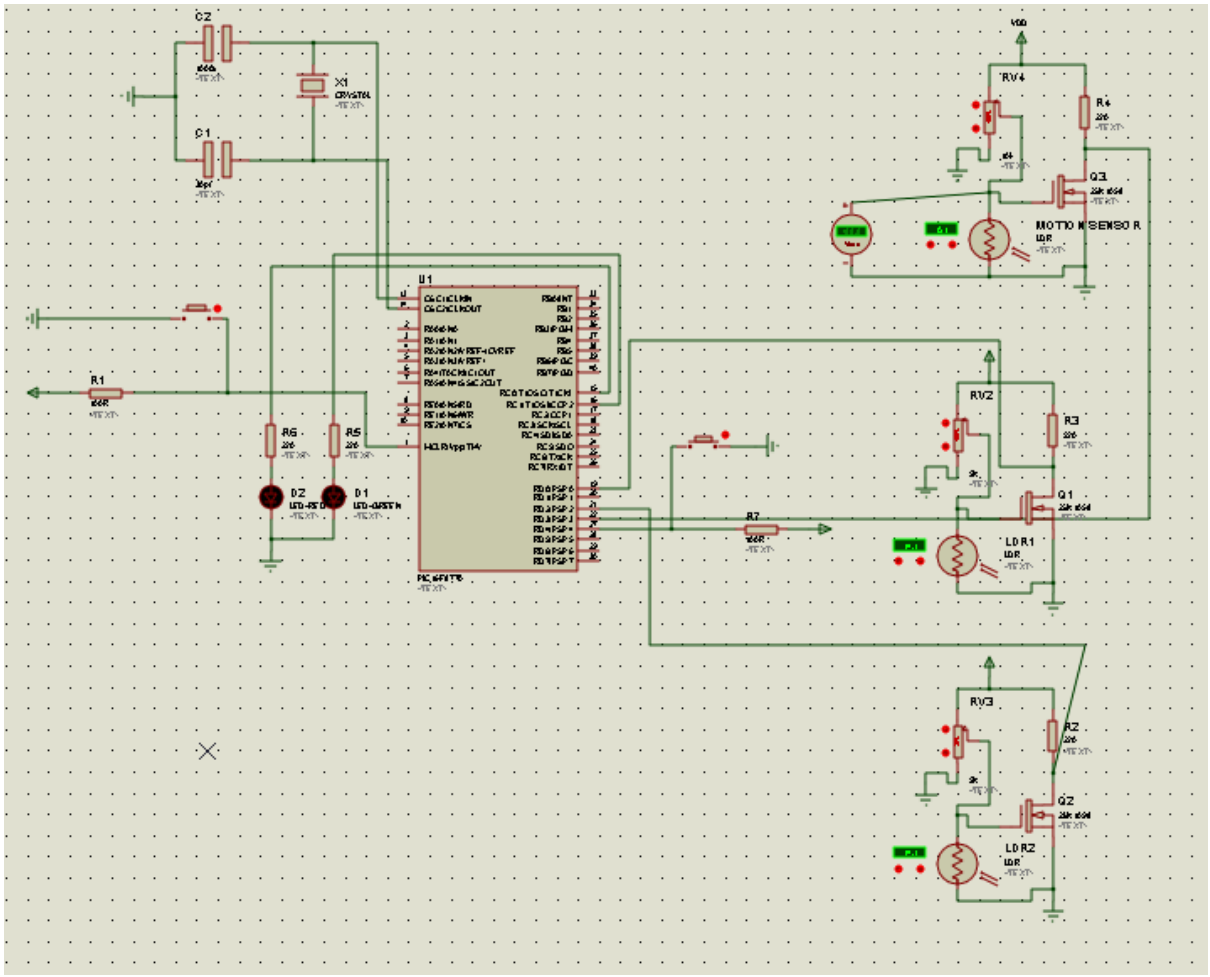

Figure 13 Schematic of Circuit with Sensors

This is the schematic design of the circuit we were able to implement in hardware.

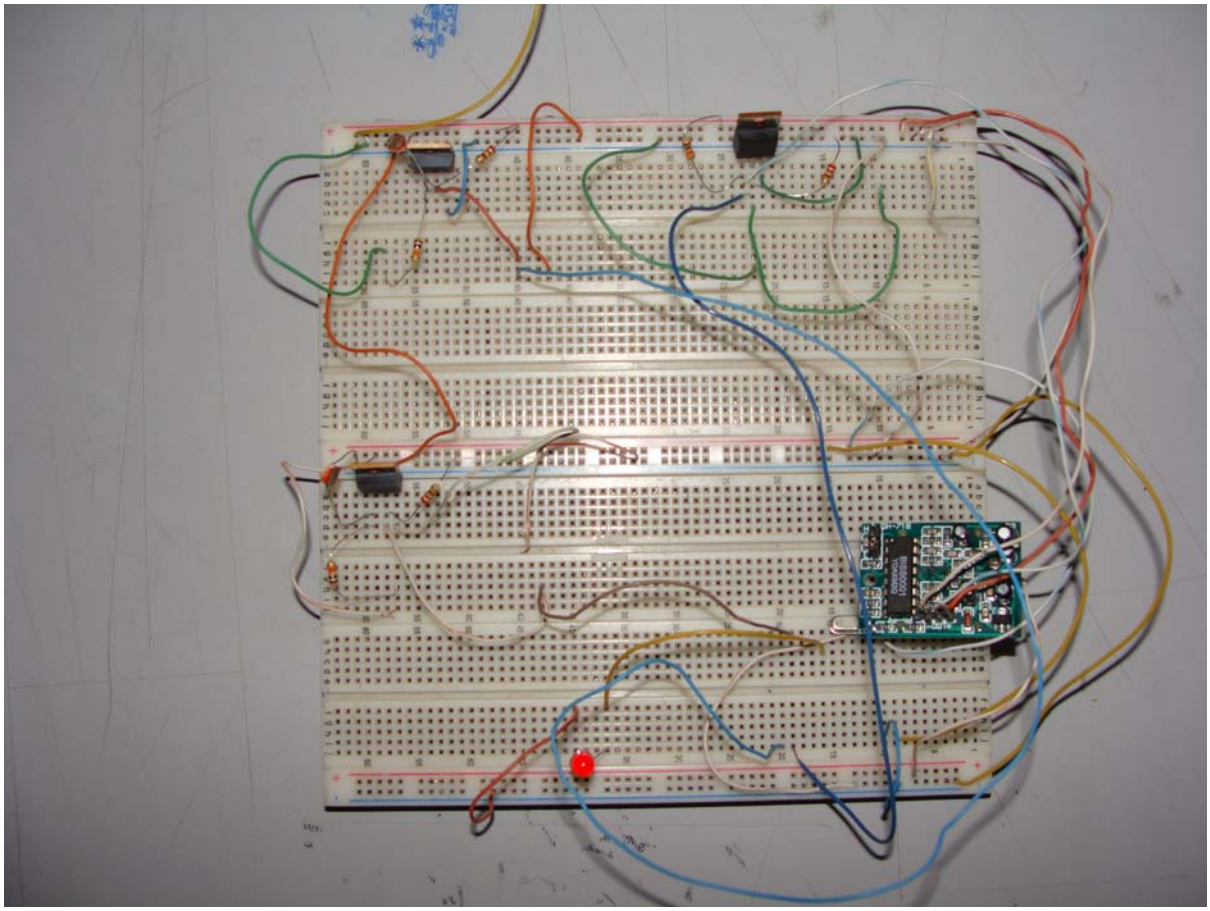

Figure 14 Circuit implemented in hardware

Future Work

Our system is designed to be highly flexible and customizable. The circuit we have shown in simulation is a very basic setup to develop and demonstrate all the functionalities of our design. The system can easily be expanded to have many more sensors thus significantly increasing is abilities. Adding more lasers will allow us to cover a greater area; more pressure sensors will allow us to guard multiple objects.

To increase functionality and security we may want to connect password operated electronic locks. The microcontroller we used is quite capable of even more functions; conversely we can use smaller less powerful microcontroller to make the system even more inexpensive. The flexibility we have with our design is enormous and thus it can be customized for specific situations and needs.

Conclusions

Through this project we believe we have developed something that has practical use in this country. Working on this project has given us the opportunity to put into practice all that we have we learned in our courses so far. It gave us an opportunity to test our skills and challenge ourselves. We have gained valuable insight into our field of study and we have learned a few new skills beyond our field. And although we were unable to reach the goal set for us, we have done the best we could within the timeframe and budget available to us.

Appendix

Code

```
//\\ for password storage and checking

float a = 0;            //\\ for storing the int value of a key press

int b = 0;              //\\ used as the flag to detect to if the (-) button has been pressed

int g = 0;

long h = 0;

long i = 0;             //\\ storing entries

long j = 0;

long k = 0;

long l = 0;

long m = 0;

int x = 1234;

unsigned short int cntr2 = 0;  //\\ second counter, used to keep track of the total number of key presses.

unsigned short int cntr3 = 5;  //\\ third counter, set to 5 since only 5 key presses are necessary



// LCD module connections

sbit LCD_RS at RB6_bit;

sbit LCD_EN at RB7_bit;

sbit LCD_D4 at RB2_bit;

sbit LCD_D5 at RB3_bit;

sbit LCD_D6 at RB4_bit;

sbit LCD_D7 at RB5_bit;
```

```
sbit LCD_RS_Direction at TRISB6_bit;
sbit LCD_EN_Direction at TRISB7_bit;
sbit LCD_D4_Direction at TRISB2_bit;
sbit LCD_D5_Direction at TRISB3_bit;
sbit LCD_D6_Direction at TRISB4_bit;
sbit LCD_D7_Direction at TRISB5_bit;
// End LCD module connections

//\\ define LED pins for output
#define LED1 Rb0_bit                //\\ blue led with buzzer
#define LED2 Rb1_bit                //\\ green led

//\\ define LED pin Direction
#define LED1_TRIS  TRISb0_bit
#define LED2_TRIS  TRISb1_bit

//\\ start of sensor input section
//\\ define LDR pin Direction
#define LDR1_TRIS  TRISA0_bit
#define LDR2_TRIS  TRISA1_bit
#define LDR3_TRIS  TRISA2_bit

//\\ define LDR pins for input
#define LDR1 RA0_bit
#define LDR2 RA1_bit
#define LDR3 RA2_bit
```

```
//\\ define pressure sensor pin for input
#define PP1 RA3_bit

//\\ define pressure sensor pin dirention
#define PP1_TRIS  TRISA3_bit
//\\ end of sensor input section


//\\ for use in keypad module
unsigned short kp, cntr, oldstate = 0;
char txt[6];
// Keypad module connections
char  keypadPort at PORTC;
// End Keypad module connections



void main(){

  cntr = 0;                       // Reset counter
  Keypad_Init();                  // Initialize Keypad
  Lcd_Init();                     // Initialize LCD
  Lcd_Cmd(_LCD_CLEAR);            // Clear display
  Lcd_Out_Cp("Keypad Disabled");
  kp = 0;                         // Reset key code variable
  cntr3 = 5;
```

```
 cntr = 0;


 //\\ make LDR1, LDR2, LDR3 and PP1 pins as input
 LDR1_TRIS = 1;
 LDR2_TRIS = 1;
 LDR3_TRIS = 1;
 PP1_TRIS = 1;


 //\\ make LED1 and LED2 pins as output
 LED1_TRIS = 0;
 LED2_TRIS = 0;


 // Wait for key to be pressed and released
   do                          //\\ start of first do/while loop, used to turn the keypad ON.
//\\ alarm
{ if(PORTD.B0||PORTD.B2||PORTD.B4||PORTD.B3)       //input in PORTA button 2 (RA2)
 {  PORTB.B0 = 1;  PORTB.B1 = 0;
 }
 else
 {  PORTB.B0 = 0;  PORTB.B1 = 1;
 }


     // kp = Keypad_Key_Press();         // Store key code in kp variable
     kp = Keypad_Key_Click(); }          // Store key code in kp variable
   while (kp!=13);                   //\\ keypad will only be activated after the ON button is pressed, end
of first do/while loop
   Lcd_Cmd(_LCD_CLEAR);
   Lcd_Out_Cp("Keypad Enabled");
```

```
 lcd_out(2,1, "Enter Password");



 do {                          //\\ start of second do/while loop. everything happens within this loop
 if(PORTD.B0||PORTD.B2||PORTD.B4||PORTD.B3)       //input in PORTA button 2 (RA2)
{  PORTB.B0 = 1;  PORTB.B1 = 0;
}
else
{  PORTB.B0 = 0;  PORTB.B1 = 1;
}



 kp = 0;                        //\\ Reset key code variable
 b = 0;                         //\\ setting the value of b at 0 at the start of the loop
   // Wait for key to be pressed and released
 do   {                         //\\ start of third do/while loop
//\\ alarm
  if(PORTD.B0||PORTD.B2||PORTD.B4||PORTD.B3)       //input in PORTA button 2 (RA2)
{  PORTB.B0 = 1;  PORTB.B1 = 0;
}
else
{  PORTB.B0 = 0;  PORTB.B1 = 1;
}
   //kp = Keypad_Key_Press();         // Store key code in kp variable
   kp = Keypad_Key_Click();}          // Store key code in kp variable
   while (!kp);                    //\\ end of third do/while loop
```

```
  // Prepare value for output, transform key to it's ASCII value
  switch (kp) {

    case  1: kp = 49;  cntr3--; a = 1; b = 0; break; // 1        // Uncomment this block for keypad4x4
    case  2: kp = 50;  cntr3--; a = 2; b = 0; break; // 2        //\\ value of b is 0 for all buttons escept (-)
    case  3: kp = 51;  cntr3--; a = 3; b = 0; break; // 3
    case  4: kp = 65;  cntr3--; b = 1; break; // A      //\\ value of b is 1 only for the (-) button
    case  5: kp = 52;  cntr3--; a = 4; b = 0; break; // 4
    case  6: kp = 53;  cntr3--; a = 5; b = 0; break; // 5
    case  7: kp = 54;  cntr3--; a = 6; b = 0; break; // 6
    case  8: kp = 66;  cntr3--; b = 0; break; // B
    case  9: kp = 55;  cntr3--; a = 7; b = 0; break; // 7
    case 10: kp = 56;  cntr3--; a = 8; b = 0; break; // 8
    case 11: kp = 57;  cntr3--; a = 9; b = 0; break; // 9
    case 12: kp = 67;  cntr3--; b = 0; break; // C
    case 13: kp = 42;  cntr3--; b = 0; break; // */ON     //\\ enable button
    case 14: kp = 48;  cntr3--; a = 0; b = 0; break; // 0
    case 15: kp = 35;  cntr3--; b = 0; break; // #
    case 16: kp = 68;  cntr3--; b = 0; break; // D

  }

  if (b != 1 && cntr3 != 0){                    //\\ check if (-) has been pressed; if not, execute the next
lines of code.

    switch (cntr3){                  //\\ stores the entries in the appropriate int according to the counter
value
```

```
case 1: j =  a*1; cntr3--; break;

case 2: k =  a*10; break;

case 3: l =  a*100; break;

case 4: m =  a*1000; break;

          }

h =  j + k + l + m;              //\\ adds up all the ints to get the final number


lcd_out_cp("#");




                  }



else { }

if (cntr3 == 0){

if (h == x){

lcd_cmd(_lcd_clear);

lcd_out_cp("Password Correct");


trisd = 1;

portd=0x00;



cntr = 0;

cntr2 = 0;                    //\\ Reset cntr2

cntr3 = 5;                    //\\ Reset cntr3

i = 0;
```

```
j = 0;
k = 0;
l = 0;
m = 0;
b = 0;                          //\\ reset b
kp = 0;                          //\\ reset kp
h = 0;
break;                          //\\ reset h
          }
else {
lcd_cmd(_lcd_clear);
lcd_out(1,1, "Re-enter Password");
cntr++;



cntr2 = 0;                       //\\ Reset cntr2
cntr3 = 5;                       //\\ Reset cntr3
i = 0;                          //\\ reset i
j = 0;
k = 0;
l = 0;
m = 0;
b = 0;                          //\\ reset b
kp = 0;                          //\\ reset kp
h = 0;                          //\\ reset h
     }
    }
```

```
    if (cntr == 3){                  //\\ to check if the wrong password has been entered 3 times
   LED1 = 1;                        //\\ sound alarm if so
   lcd_out(1,1,"Wrong Password");
    break;
              }
  } while (1);                       //\\ end of second do/while loop. infinite loop.
}
```

References

"microcontroller". Wikipedia. 2012 wikipedia, The Free Encyclopaedia 26 Nov 2012 <http://www.wikipedia.org/microcontroller.html>.

"pic16f877a". 2012 alldatasheet, All Data Sheet 1 Dec 2012 http://www.alldatasheet.com/datasheet-pdf/pdf/82338/MICROCHIP/PIC16F877A.html

"pic16f877a datasheet". Microchip. 2012 microchip. Microchip 1Nov 2012 *ww1.microchip.com/downloads/en/devicedoc/39582b.pdf*

http://www.8051projects.net/keypad-interfacing/introduction.php