Faculty of Science + Engineering

Manchester Metropolitan University

Intruder Alarm System

Name	Mohamed Alsubaie
MMU ID	09562211
Supervisor	Pr. Nicholas Bowring
Subject	Electronic Engineering
Unit code	64ET3516
Course	BEng (Hons) Computer and Communication Engineering

Acknowledgment

I would like to firstly thank my supervisor Pr. Nicholas Bowring for his support throughout this unit. Special thanks for Mr. Dean O’Reilly and other Academic staffs for their assistance and guidance in the laboratory works. Thanks to all technicians staffs for providing the require component and making the PCBs. Finally Thanks to my parents, family and friends for their help, encouragement and guidance, who without I would not be where I am today.

Report Outline

The report will start with a small introduction about the intruder alarm system as well as explaining the aim and objectives for the project. Then, the architecture of the complete design is going to be discussed in a block diagram. All the important hardware used in this project will be described in the Hardware Discretion section. This section will also cover important details about both PIR and Microphone Circuits, such as amplification stage, filter stage, and signal conditioning stage. There will also be a small section that will cover the PCBs design for this project. The next section will cover the software description of the system as well as including some flow chart for the programme. The report will end up with a conclusion and a further work sections, which are going to summarise the main findings and provide some suggestion to improve the project.

Table of Content

1. Introduction

In the past few decades, security peoples were hired to guard the areas. House owners were invested descent amount for in order to provide security. There is nothing equal to human beings. However, as the technology develops electronics security systems gained popularity among the people because of various factors like fastest and easiest method of security, better security and large coverage area (Home Security Guru, 2012). This project is an attempt to create a simple intruder alarm system, which detects movement and sound levels being activated by either human voice or motion. It will start with designing the PIR and Microphone circuits, followed by choosing an appropriate amplification and filtering stages. Then, it is going to cover some methods of signal conditioning stages in order to provide an output of 5V, which is compatible with microcontroller. After that, there will be some challenges to design good software for the overall system, as well as connecting it to the internet.

2. Aim

- The aim of this project is to design, build and develop an intruder alarm system.

3. Objectives

- Understand the characteristics of amplifiers and filters.
- Design and build a PIR circuit.
- Design and build a Microphone circuit.
- Signal conditioning stage to be compatible with microcontroller 5V.
- Write good programmes for both PIC and Arduino microcontrollers.
- Control the system over internet use Ethernet Shield.
- Make a good user friendly interface.
- Develop PCB design and soldering skills.

4. Specification

The system requirements are given below:

- ±15V power supply voltage for op-amps.
- 5V supply is required for microcontrollers and other components.
- Internet connection RJ45 cable for the Ethernet Shield.

5. System Architecture

The block diagram of the complete design is shown in figure-1. The system can be divided into two parts. The first part is all about controlling the inputs and outputs of the system. As it illustrated, the PIC microcontroller “PIC18F2420” is going to read the outputs of PIR circuit, microphone circuit and keypad, and controls the outputs of LCD, buzzer and LEDs. Moreover, the second part is about checking and controlling the system through the internet. A web page has been created by programming Arduino Uno Board and Ethernet Shield. In order to allows the web page to control the system, both microcontrollers have been connected together and programmed to the understand each other. The full circuit is included on appendix-1.

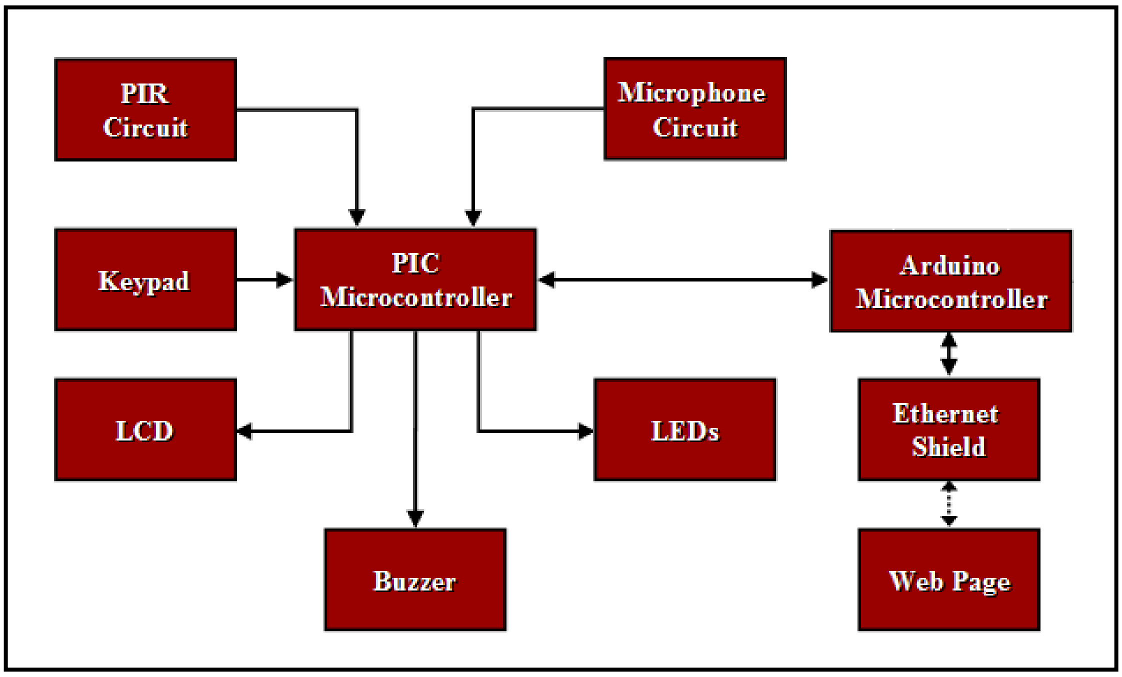


Figure-1: Block diagram of the security alarm system.

6. Hardware Description

6.1 Microcontrollers

▪ **PIC18F2420**

This family offers the advantages of all PIC18 microcontrollers. It has high computational performance at an economical price, with the addition of high-endurance, Enhanced Flash program memory. Moreover, PIC18F2420 uses nano watt technology, which can significantly reduce power consumption during operation. The microcontroller have multiple of internal oscillator, 8 use-selectable frequencies from 31 kHz to 8 MHz. Some of the important features are summarized in table-1. Programming a PIC microcontroller in C language requires using MPLAB IDE with C18 library. After compiling the programme, the C code will converted into machine language (hex file). In order to upload this file to the microcontroller, a PIC programmer is required. In this project the programmer is PICkit3, see appendex-4 for PIC18f2420 pins diagram and appendex-5 for PICkit3 pins out.

Table-1: Summary of important PIC18F2420 features (Arduino, 2012).

Device	Program Memory		Data Memory		I/O	10-Bit A/D (ch)	CCP/ ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I^2C™			
PIC18F2420	16K	8192	768	256	25	10	2/0	Y	Y	1	2	1/3

▪ Arduino Uno

The Arduino Uno is a microcontroller board that contains everything needed to support the microcontroller. It is made up of an Atmel AVR Microprocessor (which is ATmega328), a 5 volt linear regulator, a power jack, a USB connection, an ICSP header, a reset button, a 16 MHz crystal oscillator which is enable the Arduino to operate at the correct speed by sending the right time pulses with specified frequency, and an Atmega8U2 that programmed as a USB-to serial converter (McRoberts, 2010). Some of the important features are summarized in table-2 below. Arduino boards can be easily programmed using Arduino Integrated Development Environment. It has many libraries and functions that will help users to do incredible stuffs (Arduino, 2013). After compiling the programme, a hex file will be generated and it can be uploaded by connecting the Arduino board and click Build icon.

Table-2: Summary of important Arduino Uno features (Arduino, 2013).

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage(recommended)	7-12V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analogue Input Pins	6
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

6.2 PIR Circuit

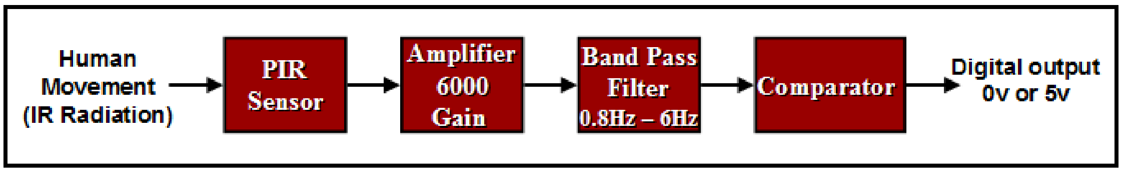


Figure-2: Block diagram of the PIR circuit.

▪ PIR Sensor

The block diagram in figure-2 describes the process of detecting movement being activated by human. It starts with Passive Infrared Sensor, which is an electronic sensor that measures the change in temperature radiation omitted form human body in its field of view. This PIR sensor uses a pyroelectric sensing element that has a very low level analogue signal output. It has an unstable AC signal from 1mVp-p to 2mVp-p with high frequency noise component around 10mVp-p. This signal rides on a DC component of 1V that will vary with temperature. In order to increase the detection area, a Fresnel lens has been placed on the top of the PIR. It helps to focus the infrared radiation by divides the whole area into different zone, any movement between zones will causes a change in the IR (Zilog, 2010).

▪ Amplifier and Filter

Since the output signal of the PIR sensor is very small, it needs to be well amplified to be use by other components to make decisions and act upon them. Therefore, a two stage amplifier with a total gain of 6000 has been used. Moreover, the PIR signal is required to have a tight band pass filter stage around 0.1Hz to 10Hz to reduce high frequency noise and strip the DC element that the signal rides on (Zilog, 2010). In this project the circuit that been used has a band pass filter of 0.8Hz to 6Hz.

▪ Signal Conditioning

The last stage is the comparator stage that gives the output voltage of either 5V for logic one or 0V for logic zero depending on the threshold voltage across it input, see figure-3. In this project, the comparator has been provided with a 10kΩ potentiometer, so the threshold voltage can be change and adjusted any time. The comparator output can be used together with a pull-up resistor by a microcontroller to make decisions and act upon it.

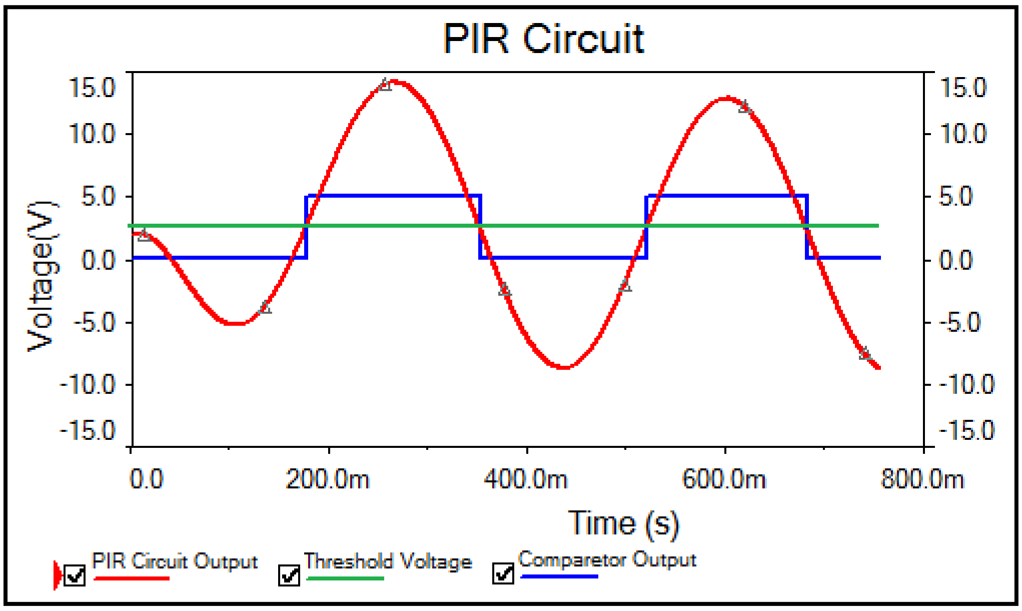


Figure-3: The output of the PIR circuit.

6.3 Microphone Circuit

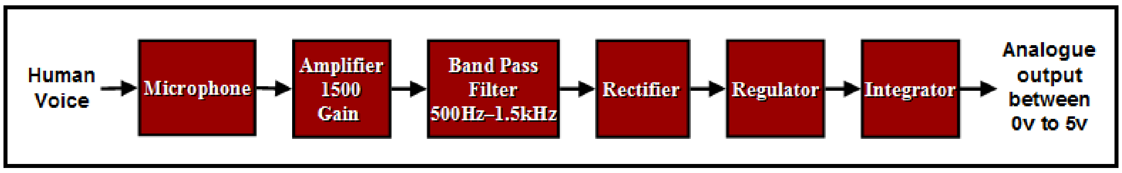


Figure-4: Block diagram of the microphone circuit.

▪ Condenser Microphone

The block diagram in figure-4 describes the process of monitoring the sound levels being activated by human voice. All microphones are transducers that convert sound pressure waves into an electrical signal. Microphone is similar to capacitor. It made of a dielectric material that holds a permanent charge. Vibrates effects will causes a change in the internal capacitance and an electrical signal will produced. In order to pick up a voice signal a pull up resistor of 4.7kΩ is used to determine the output impedance on the microphone. Moreover, the output signal has filtering capacitor of 0.1uF which is responsible to block any DC voltage bias from the output.

▪ Amplifier and Filter

Since the output signal of the microphone is very small, a one stage amplifier with a gain of 1500 has been used to amplify this signal. As the amplifier gain increase, the detection area will also increase. Moreover, the microphone has been tested with different animal and human sounds. For each voice, four readings have been recorded using oscilloscope. The bar chart in appendix-6 shows all the recorded results for the tested sounds. From these results, the suitable values for the band pass filter to remove non human voice have been selected to be from 500Hz to 1.5kHz.

▪ **Signal Conditioning**

The microphone circuit provide an output signal of ±15V. In order to make it suitable for microcontrollers, this signal needs to go through the following stages:

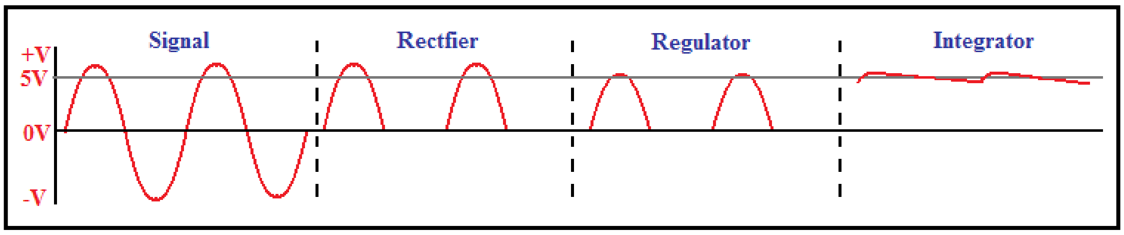


Figure-5: Signal conditioning stages for the microphone circuit

Rectifier: Removing the negative signal using a diode.
Regulator: regulate the voltage to maximum of 5V using zener diode.
Integrator: by using an RC network, the output signal is going to be smooth nearly DC output and it will be suitable to use with microcontrollers.

In this case, this method is better than using a comparator, because having an analogue output for the microphone circuit will provides the opportunity to improve ignoring the non-human voice through programming.

6.4 Ethernet Shield

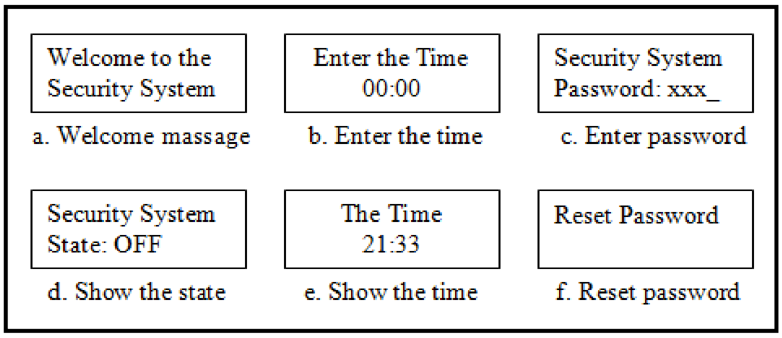
The Arduino Ethernet shield enables your Arduino to send and receive data from anywhere in the world with an internet connection just by plugging it onto the Arduino board, connecting it to the network (RJ45 cable) and following few simple instructions. All hardware, software and documents are freely available and open source (Arduino, 2013). This shield is based on the Wiznet W5100ethernet chip, which will provide an IP address stack capable of both TCP and UDP. It is very easy to programme this shield using the Ethernet library , which is available on Arduino IDE. This shield has been used in this project to allow the user to check the system state and to have the ability to turn on or off the system through the internet, using mobile phones, tablet, laptop, PC or any other devices that has internet connection (McRoberts, 2010).

6.5 Keypad

Matrix keypads are well known for their simple architecture and ease of interfacing with any microcontroller. In this project, a 12 buttons 3x4 matrix keypad has been used to give the opportunity for the user to select the wanted mode and to enter the time or the password.

6.6 Liquid Crystal Display

In this project, a Liquid Crystal Display have been used to provide a better user interface by showing massages on the display, which is an easer way to make the users follow the right instructions, see figure-6 below. This LCD consisting of to 2 lines, each line can handle 16 characters. The connection could be quite large and required more pins. However, it could be minimised in two steps. The first step is to connect the Read/ Write pin to the ground because it will not required. The second step is to use 4 data lines instead of 8 lines, and send the data twice.



Welcome to the Security System	Enter the Time 00:00	Security System Password: xxx_
a. Welcome massage	b. Enter the time	c. Enter password
Security System State: OFF	The Time 21:33	Reset Password
d. Show the state	e. Show the time	f. Reset password

Figure-6: Some important message that will display on the screen.

6.7 Buzzer and LEDs

A buzzer, which is an audio signalling device, has been used in this project to give an alarm sound when the PIR or microphone detects human movement or sound. Since buzzer requires square wave to make sounds, a PWM signal from the microcontroller will be used.

A light-emitting diode "LED" is a semiconductor light source. There are many different colours for LEDs. They are used in this project to provide a good user interface. Three colours has been selected Red, Yellow and Green, each led has it own current limited resistor, see figure-9.

6.8 Power Supply

Powering on the system requires +15V -15V and 5V. A variable power supply from the lab can be use. However, reducing the gain so it becomes suitable with working at 9V will give the opportunity to use two 9V batteries and one voltage regulator "5V" to power on the circuit. The power supply connection is available on appendix-8.

7. PCB Design

The PCB design for PIR and microphone circuit has been done using Multisim and Ultiboard soft-wares, while the PIC circuit has been done through Eagle PCB and POV for 3D image. The bottom layers of the PCBs can be found in appendix-3. In all PCBs solid ground panel has been used, which will provides continuous low impedance path for return current. In addition, the power supplies are having thicker tracks and connected in a star configuration. The figure below shows the top layer of final PCBs and there outputs. There was a small problem in the PIR PCB, the comparator was giving a square wave by it-self and it response get better by connecting an oscilloscope to it input. This means that the comparator cannot capture the input signal, and it is necessarily to slow down the signal with a capacitor. Therefore an extra 10nF has been soldered in the top layer of the PIR PCB.

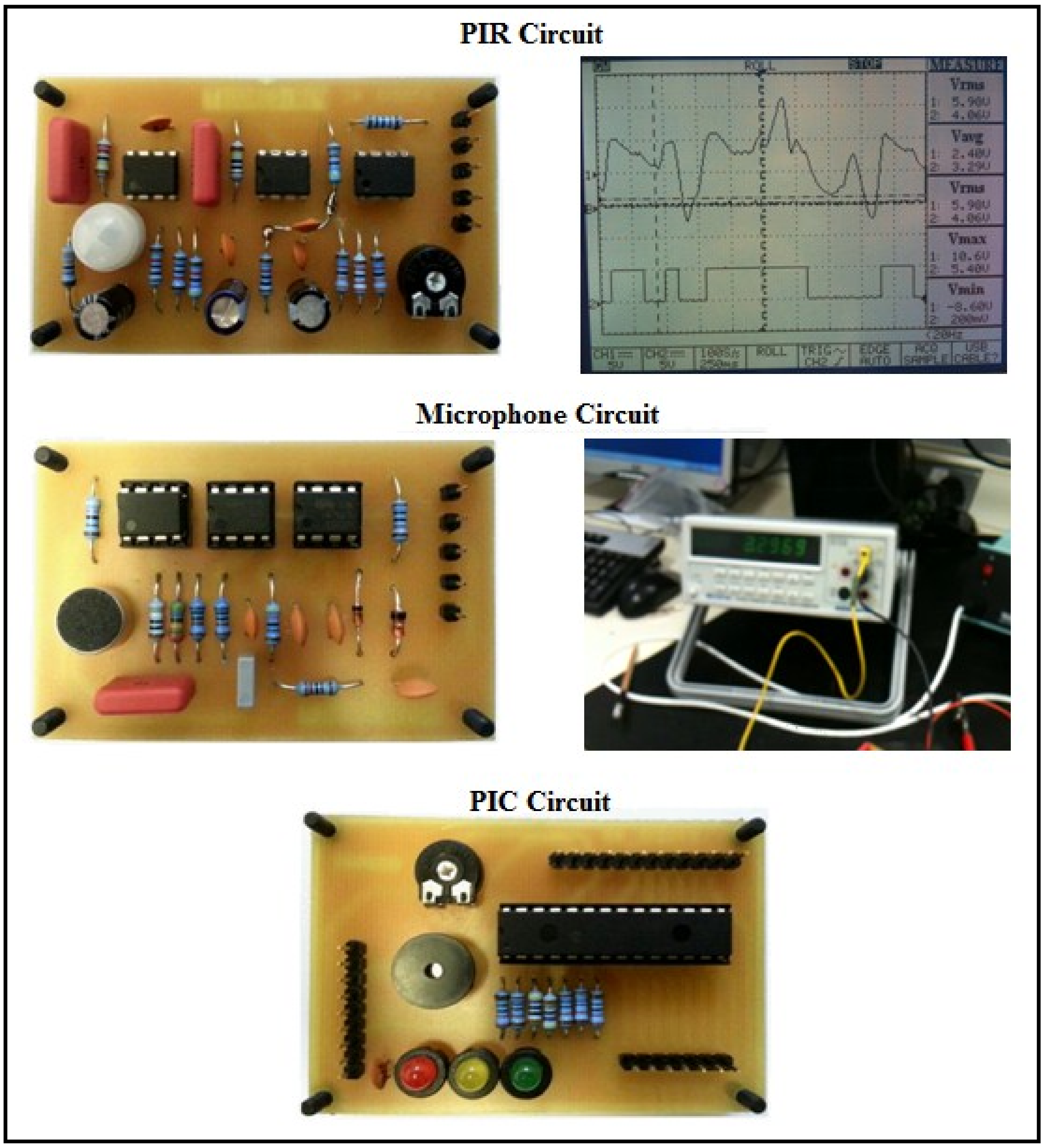


Figure-7: Final PCBs.

8. Software Description

This software will provide many features for the user. It has a 24 hour clock, password identifier and very simple instructions that can be easily followed. Moreover, this system is capable of store and show the last warning time. It also provided with a reset password option, so the password can be change any time. In addition, the software will allow the user to control and check the system through the internet. The figure below is showing the flow char of the complete system. The top one is the main programmed, while the rest are a functions that will called in the main flowchart.

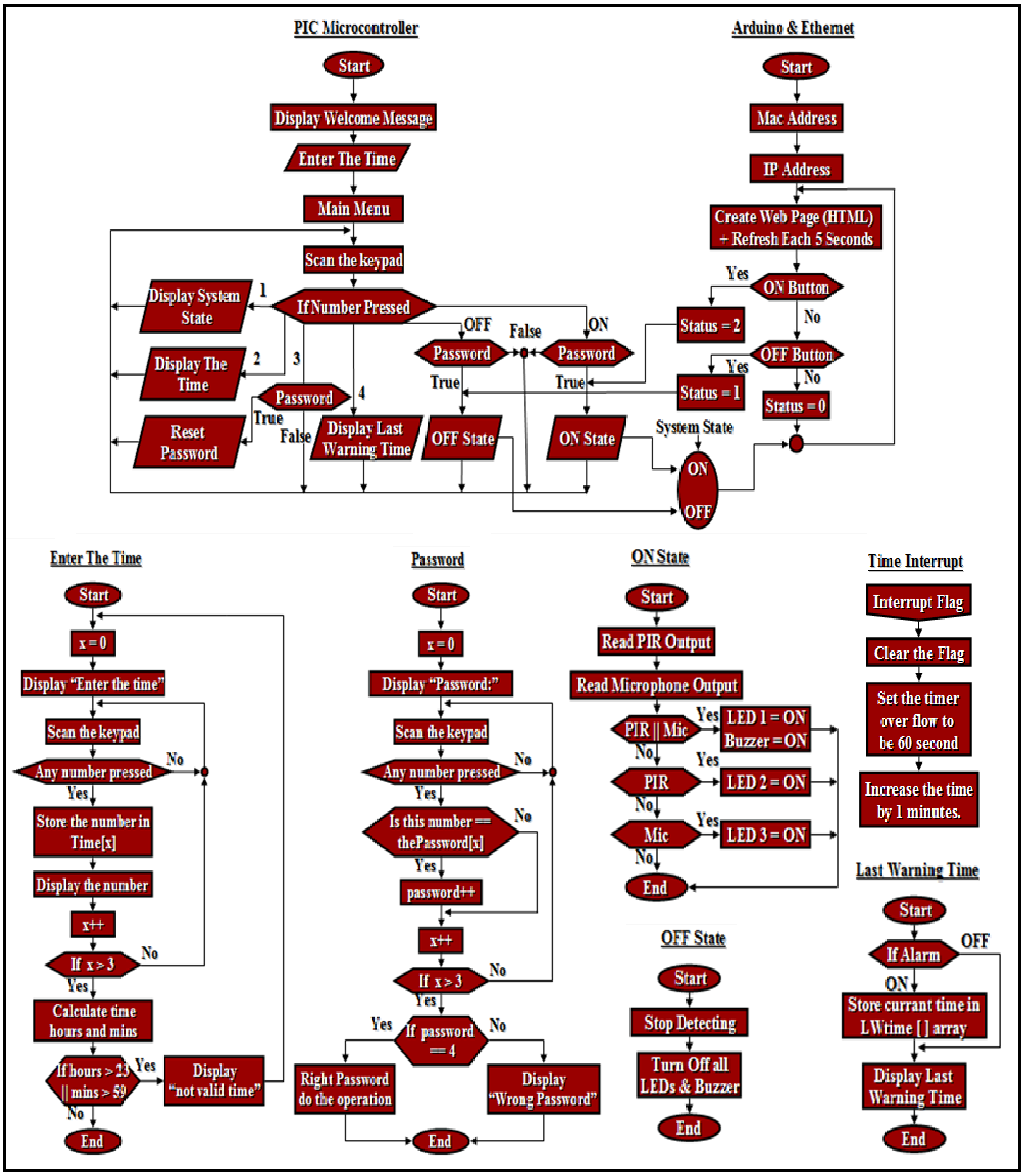


Figure-8: Flow chart of the programme.

8.1 PIC Microcontroller

The programme on the PIC microcontroller will start with showing a welcome message and then ask the user to enter the time. In this process the system will wait the user to enter 4 numbers, see “Enter The Time” flow char. From these numbers, the value of the hours and minutes will be calculated. If the entered time is not available (such as 27:80) LCD will display a not valid time massage and ask for correcting the time.

After that, the programme will display the system state, which is also the main menu, and it will also wait for the user option:

Table-3: Main menu option for the system.

Option Key	Option Name	Option Description
1	System State	Show the system state: ON, OFF or Warning!
2	Show the Time	Display the current time.
3	Reset Password	Enable user to change the password (old password is required).
4	Show Last Warning Time	Display the last warning time.
ON	ON State	Turn on the system and enable human detection (the password is required).
OFF	OFF State	Turn off the system and disable human detection (the password is required).

▪ Time Interrupt

Interrupt by TMR0 overflowing with a clock speed of 31kHz, set **TMR0H** to 0xF8 and **TMR0L** to 0x7F. The 16 bit **TMR0** overflows at 0xFFFF setting it initially to 0xF87F means it will run for 0x0780 before overflowing and interrupting approximately 1 minute. Each 60 second the programme will be interrupted to update the time values inside the programme. It will only update the time on the LCD when the user is on the “show the time” option.

▪ Password Identifier

Entering the password is done by calling **enterPassword( )** function, which is takes four integers numbers and compare them in the right sequence of the numbers inside **thePassword[ ]** array and for each match number increase integer **password** by 1. So, for the correct password integer **password** should have a total value of 4 otherwise the password is wrong. This system has a default password which is “1234”. This password can be change any time.

▪ Reset Password

In order to reset or change the password, the user will be asked for the old password. When the right password entered the system will allow the user to store the new password on **thePassword[ ]** array. In case of entering the wrong old password, the system will display a wrong pass word message to inform the user and will go back to the main menu.

▪ Scanning Keypad

The keypad is connected to PORTB because it contains internal pull-up resistors. Therefore, it is necessarily to turn on the internal pull up resistors in PORTB "**INTCON2bits.RBPU=0**". The internal pull-up resistors are only applies on PORTB inputs and they are not having any effects on pins configured as outputs (MikroElektronika, 2012).

The scan function is scanning the key pad in the following order:
RB4 low and scan RB0 to RB3 for a low to check which key had pressed 1, 4, 7 or ON.
RB5 low and scan RB0 to RB3 for a low to check which key had pressed 2, 5, 8 or 0.
RB6 low and scan RB0 to RB3 for a low to check which key had pressed 3, 6, 9 or OFF.

▪ Last Warning Time

When alarm turn on because of movement or voice detection, the system will store the warning time in an array called LW_time[0]. Then the user will be able to see this time by pressing number 4 on the main menu using the keypad.

▪ Read PIR & Microphone Output

In order to turn on the alarm, the PIC microcontroller is required to read both PIR and microphone output. Since the PIR circuit provide a digital output, reading this signal will be straightforward. Simply, by set C3 pin to be an input and connect the PIR output to it. However, reading an analogue signal from the microphone output will require an analogue to digital converter. The PIC18F2420 has an ADC of 10 bit resolution. Therefore, the converter can divide the analogue input voltage between 0v and 5v to 2^ 10 levels, which are 1024 levels from 0 to 1023.

```
ADCON0bits.CHS2 = 1;                    //select the right channel for ADC
ADCON0bits.ADON = 1;                    //turn on A/D
ADCON2 = 0b10000000;                    //right justified

ADCON0bits.GO_DONE = 1;                 //do A/D measurement
while (ADCON0bits.GO_DONE == 1);        //wait until bit = 0  measurement completed
voice = ADRESL + (ADRESH * 256);        //store the value in integer voice
```

▪ PWM

As discussed before PWM signal is needed for the buzzer to generate sound. The following code explains the steps followed to configure the Capture Compare PWM (CCP) module for this PIC.

```
T2CON = 0b00000100;             //prescaler + turn on TMR2;
PR2 = 0b00001111;               //set the right value for PR2
CCPR1L = 0b00000111;            //set duty MSB
CCP1CON = 0b00111100;           //duty lowest bits + PWM mode
```

8.2 Arduino & Ethernet Shield

The Arduino programme will start with providing the IP address and creating the web page shown in appendex-1. The web page will show the state of the system, movement detection and voice detection. It also contains two radio buttons and one submit button, which is needed to turn on or off the system. This web will automatically refresh each 5 second to update the information.

As shown in figure-9, the Arduino is has three input coming from the PIC and two outputs going to the PIC. When the ON button is submitted in the web page, the status values will be 2. This is going to turn on the ON Pin, as well as the system state. As long as the status value is equal to 2, the user will not be able to close the system using the keypad. Therefore, the Arduino programmed to give a trigger or pulse to the PIC to avoid this problem. So, after submitting ON or OFF button the status value will be 2 or 1 for short period of time and then returned automatically to 0.

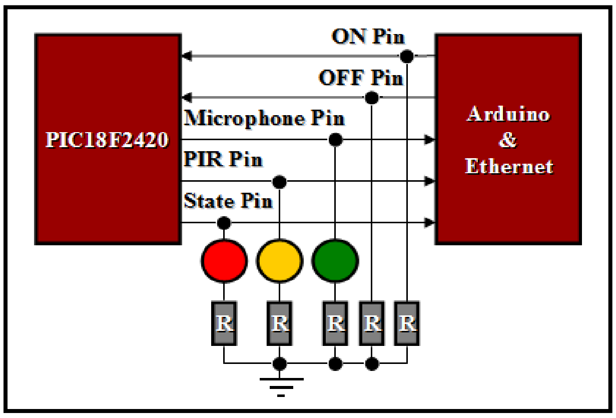


Figure- 9: Arduino and Ethernet connection.

9. Conclusion

In conclusion, the project requirement has been successfully achieved. Both PIR and Microphone circuit has been design and build on PCB. Since the output of these circuit was very small, the required to amplification and filtering stages. In order to provide these signals to a microcontroller, some signal conditioning methods have been applied. In the PIR circuit, a comparator has been use to provide a digital output 0V or 5V. While in the microphone circuit a method has been followed to provide an analogue output from 0V to 5V.

PIR and Microphone PCBs have been used together with PIC microcontroller, Arduino Uno board, Ethernet shield, keypad, LCD, Buzzer and LEDs to create an intruder alarm system. The over all system was able to detect human movement and voice and provide the user with many futures such as, good user interface, easy instruction to follow and an internet access to the system.

The outcome of this project has developed circuit design and analysis skills. This project offers an excellent knowledge about filter, amplification, and signal conditioning stage. As well as, gaining some experience with using engineering software tools for simulating circuits and making PCBs. It also provides the opportunity to apply and develop the programming skills for both PIC and Arduino microcontrollers.

10. Further work

10.1 Digital Audio Possessing

Given the slow speed of the PIC and Arduino microcontroller, it is difficult to process audio in real-time. However, with a very fast computer it will be possible to improve the project with adding Digital Signal Processing. This process starts with a high resolution ADC with specified sampling rate, in order to converts an analogue signal from the microphone to a digital signal. Most audio hardware for computers includes 16-bit converters and supports sampling rates of up to 44100Hz. A band pass filter is also needed to avoid the distortion of the frequencies higher than Nyquist frequency, which is half of the system’s sample rate. This will help to have much better human voice detection for our project.

10.2 Reset Button

Since this security alarm system has an on and off switch, it not very necessarily to have a reset button. However, a reset button can be added to the PIC circuit by placing a button from GND to the MCLR pin of the microcontroller including the same pull up resistor 4.7kΩ to VDD. This will help to reset all the registers and placing them to the start position, if the microcontroller does not behave in the right way.

10.3 Camera

As an extra feature, a camera can be added in this project, which will take a photo if the system detects human voice or movement. This photo can be uploaded into the web page or sent to the user email using the Ethernet shield.

10.4 Software Improvement

The software can be improve by adding a function that can measure the background noise, which can be activated when the microphone will located in noisy pace. Moreover, the software can also improved by adding a feature that will allow the user to select the wanted time to turn on or off the system, such as making the system operate every day between 11:00 pm to 5:00 am.

11. References

Arduino, 2013. Arduino Uno. [online] Available at: <http://arduino.cc/en/Main/ArduinoBoardUno> [Accessed on 19 March 2013].

Bates, M. P., 2011. *PIC microcontrollers: an introduction to microelectronics.* Oxford: Elsevier Ltd. p17.

Home Security Guru. 2012. Introduction to Home Security Systems: Alarms and Sensors. Available: http://www.homesecurityguru.com/introduction-to-home-security-systems-alarms-and-sensors. [Accessed on 19 March 2013].

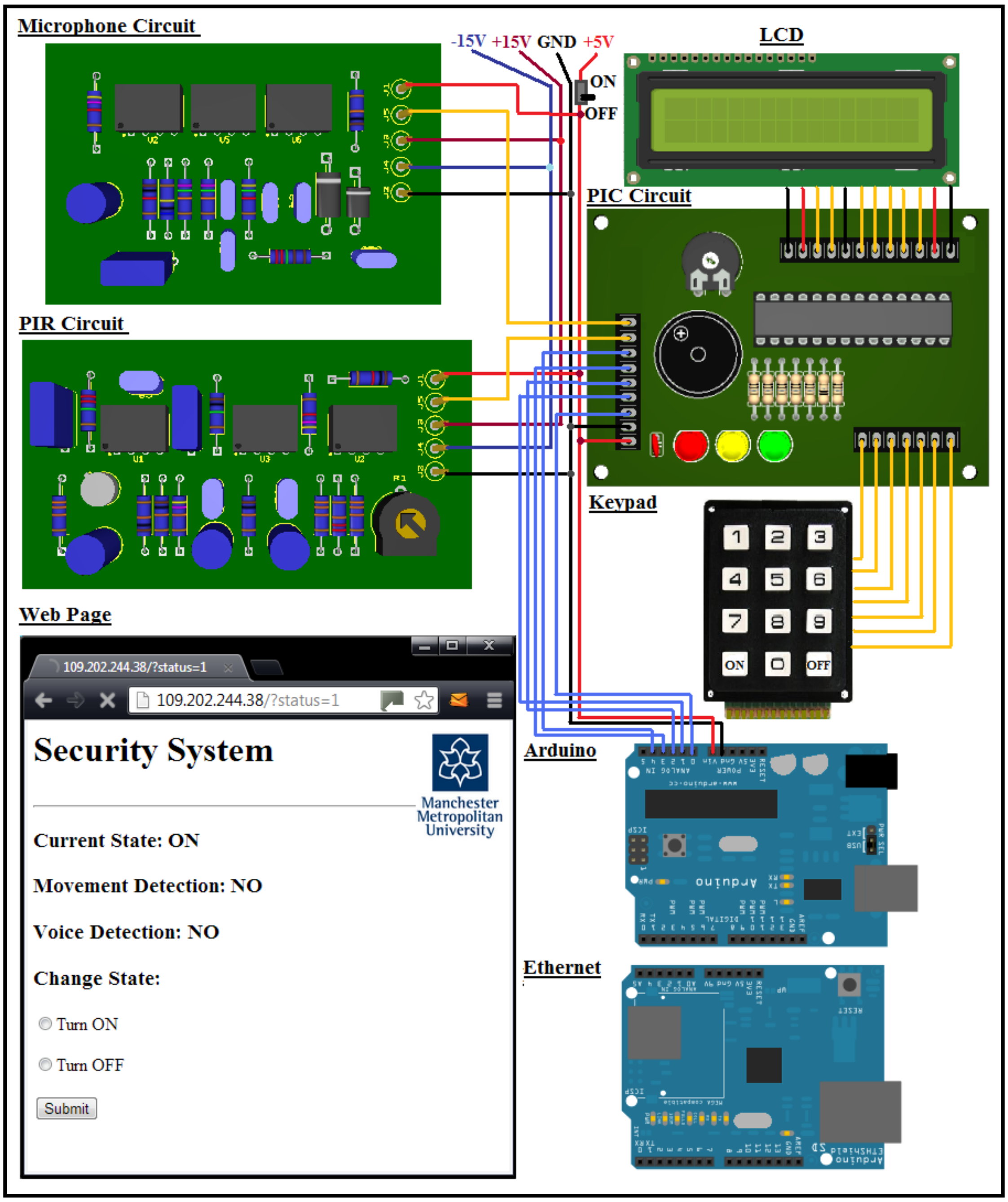
McRoberts, M., 2010. *Beginning arduino*. New York : Apress.

MikroElektronika. 2012. Inputs and outputs ports. Available: http://www.mikroe.com/eng/chapters/view/4/chapter-3-i-o-ports/. Last accessed 21st March 2012.

Zilog, 2010. *ZMOTIONth a new PIR motion detector architecture.* [pdf] WW Field Applications Manager. Available at: <http://www.zilog.com/docs/appnotes/WP0 017.pdf> [Accesseced on 19 March 2013].

12. Appendixes

Appendix-1: Full Circuit Connection.



Security System

Current State: ON

Movement Detection: NO

Voice Detection: NO

Change State:

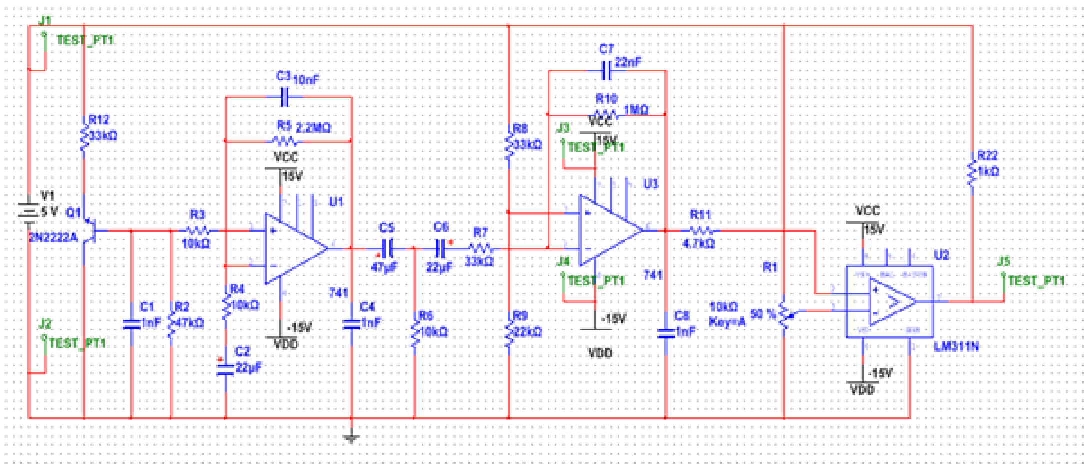
Turn ON

Turn OFF

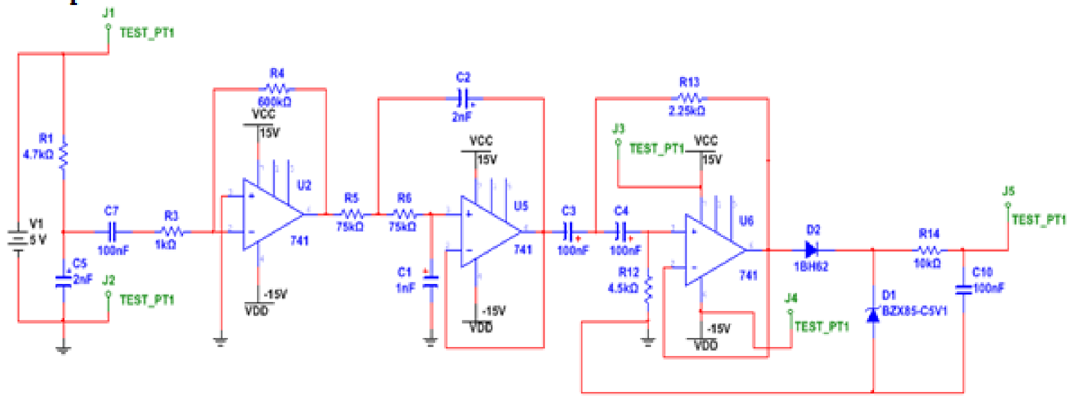
Submit

Appendex-2: Schematics Diagrams.

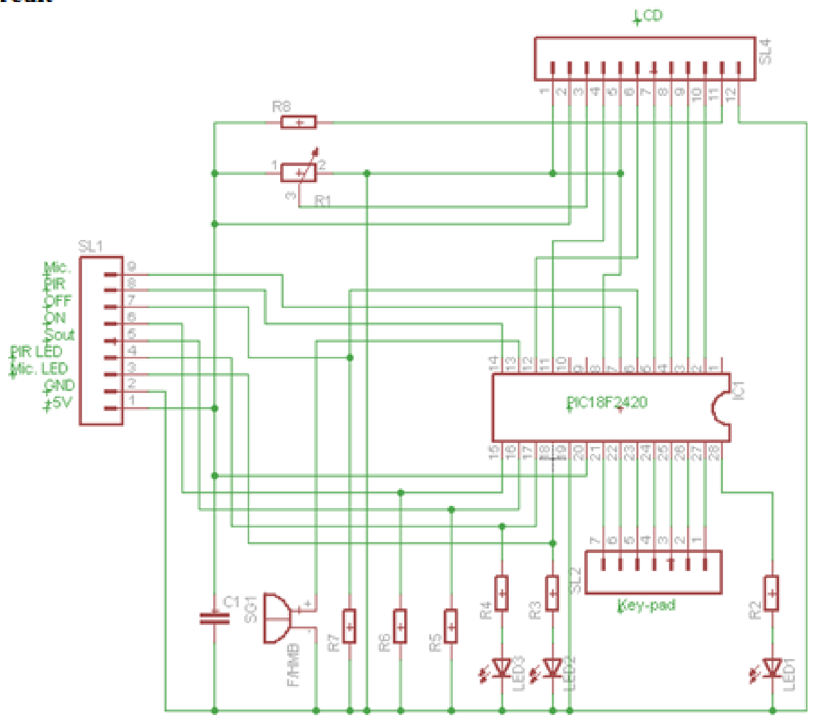
PIR Circuit



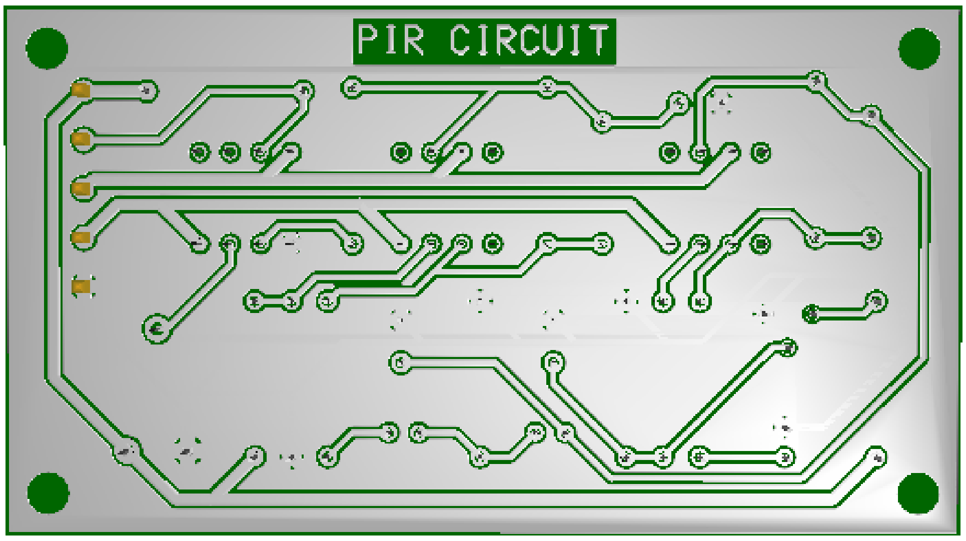
Microphone Circuit

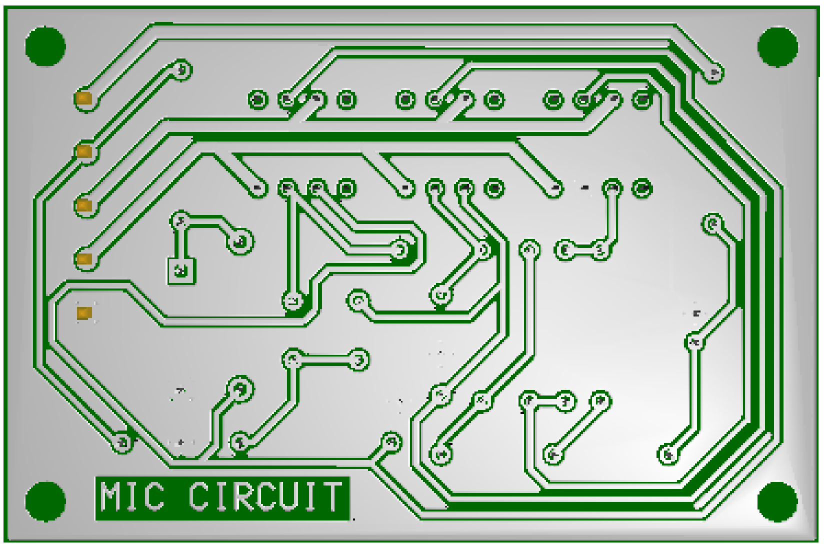


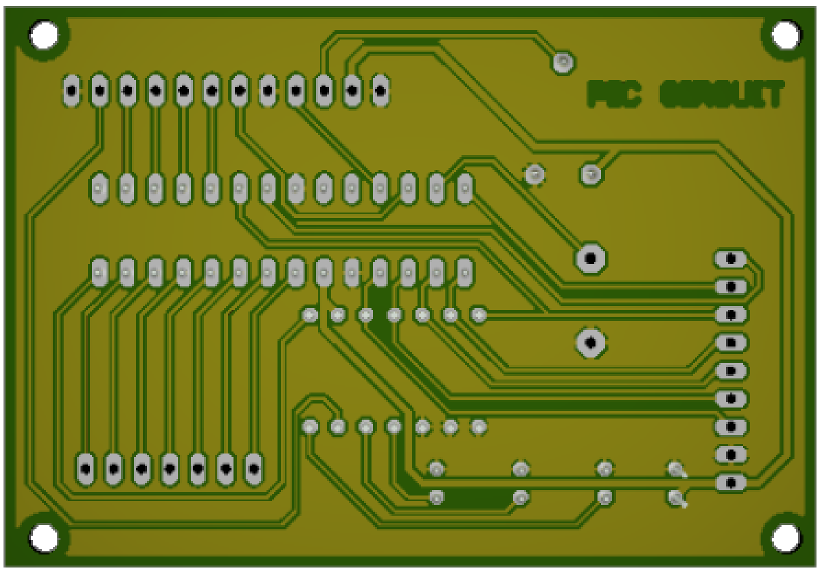
PIC Circuit



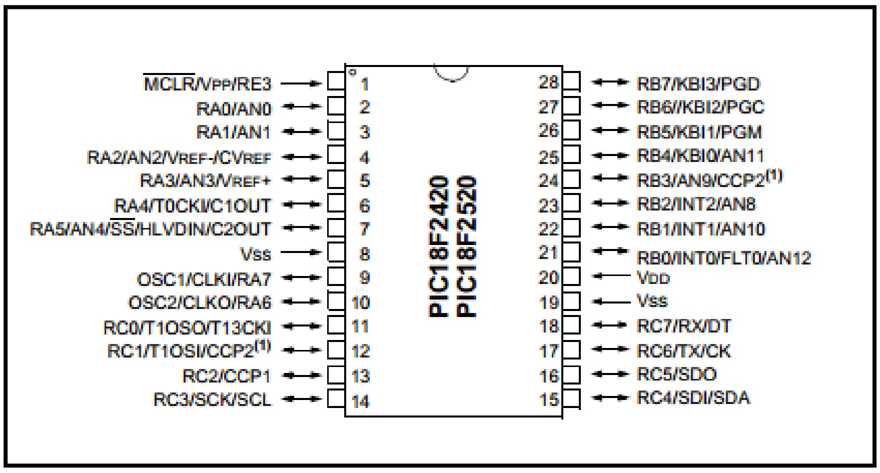
Appendix-3: PCB Bottom Layer Design.



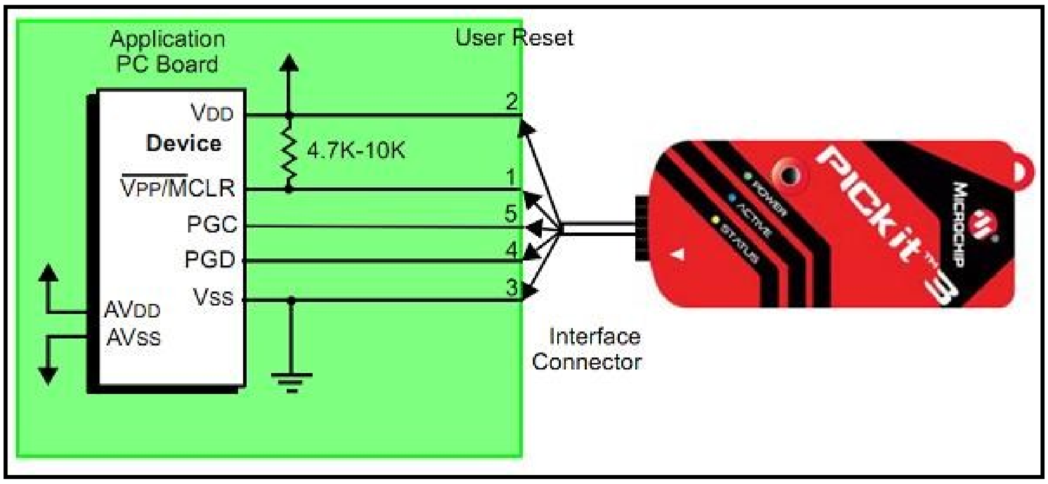




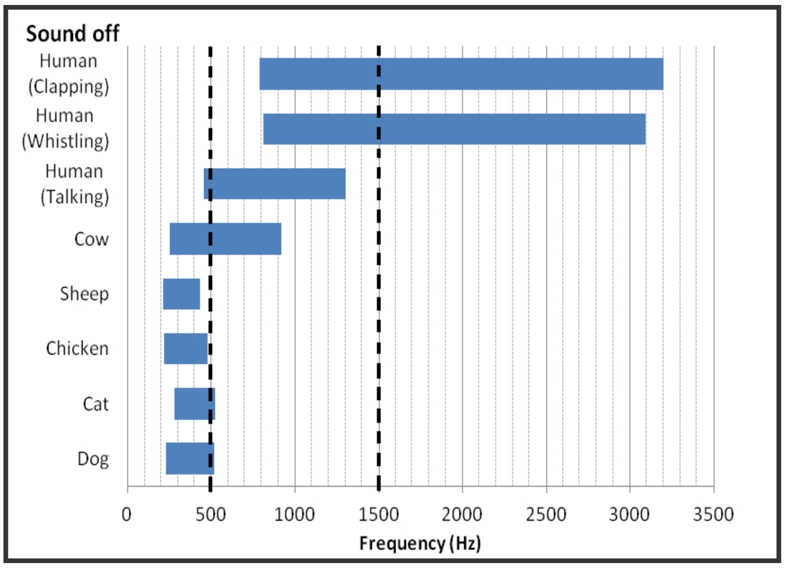
Appendix-4: PIC18F2420 Pins Diagram.



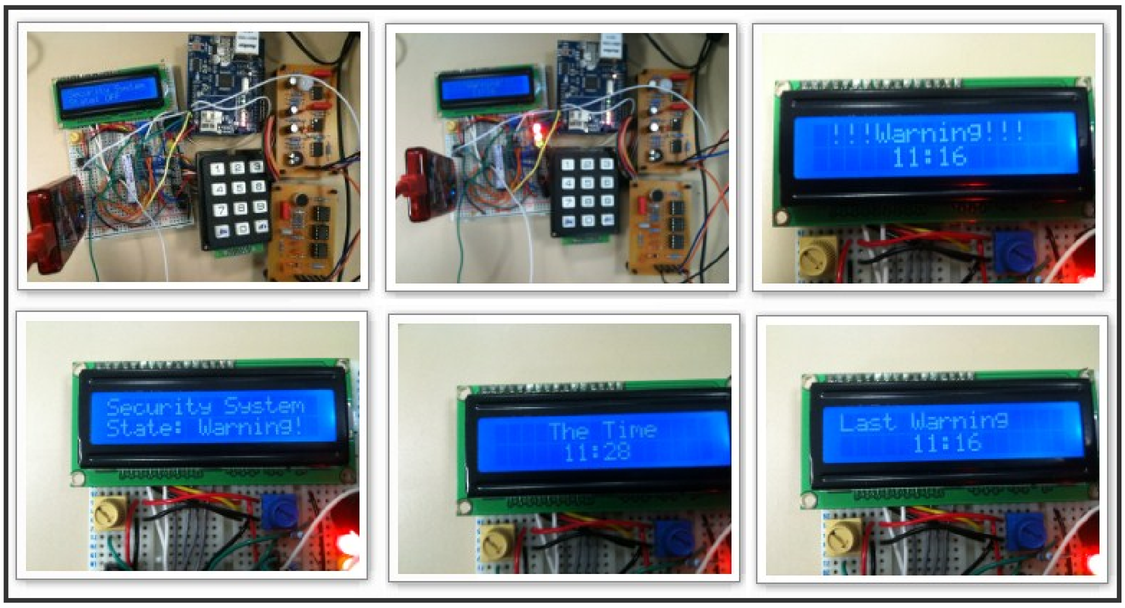
Appendix-5: PICkit3 Pins-out.

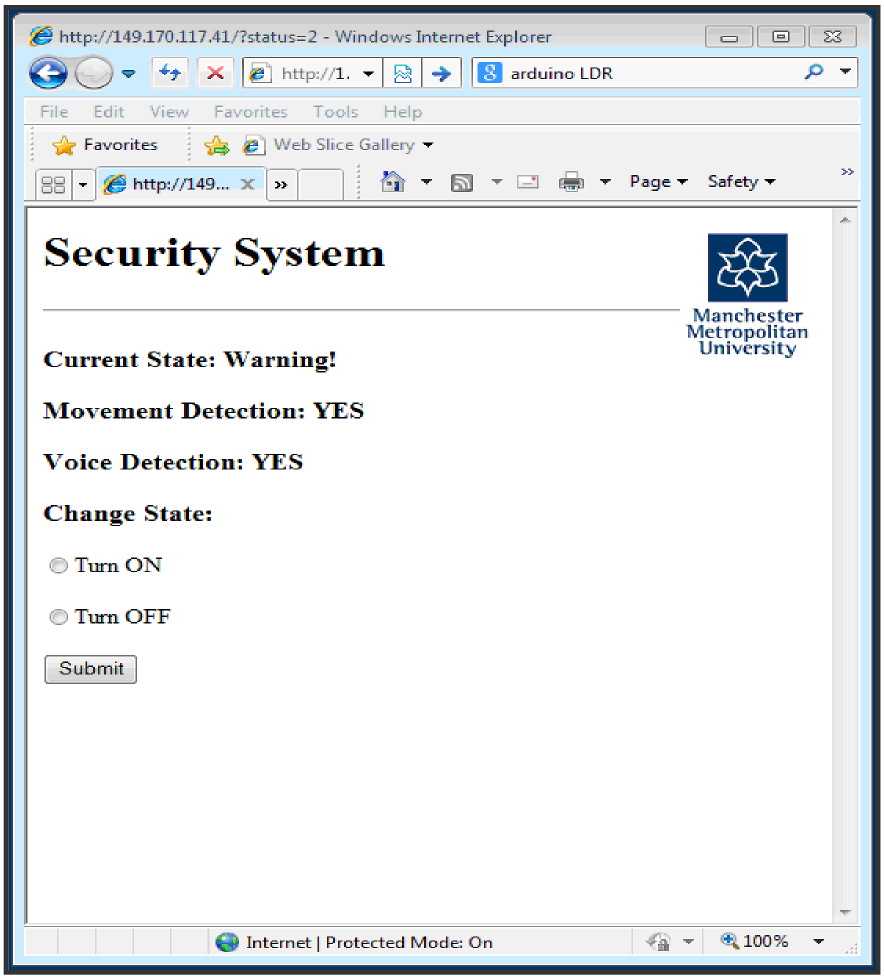


Appendix-6: Human and animal sounds results.



Appendix-7: Testing the System.




http://149.170.117.41/?status=2 - Windows Internet Explorer
arduino LDR
File Edit View Favorites Tools Help
Favorites
Web Slice Gallery
Page
Safety
Internet | Protected Mode: On
100%

Security System

Manchester Metropolitan University

Current State: Warning!

Movement Detection: YES

Voice Detection: YES

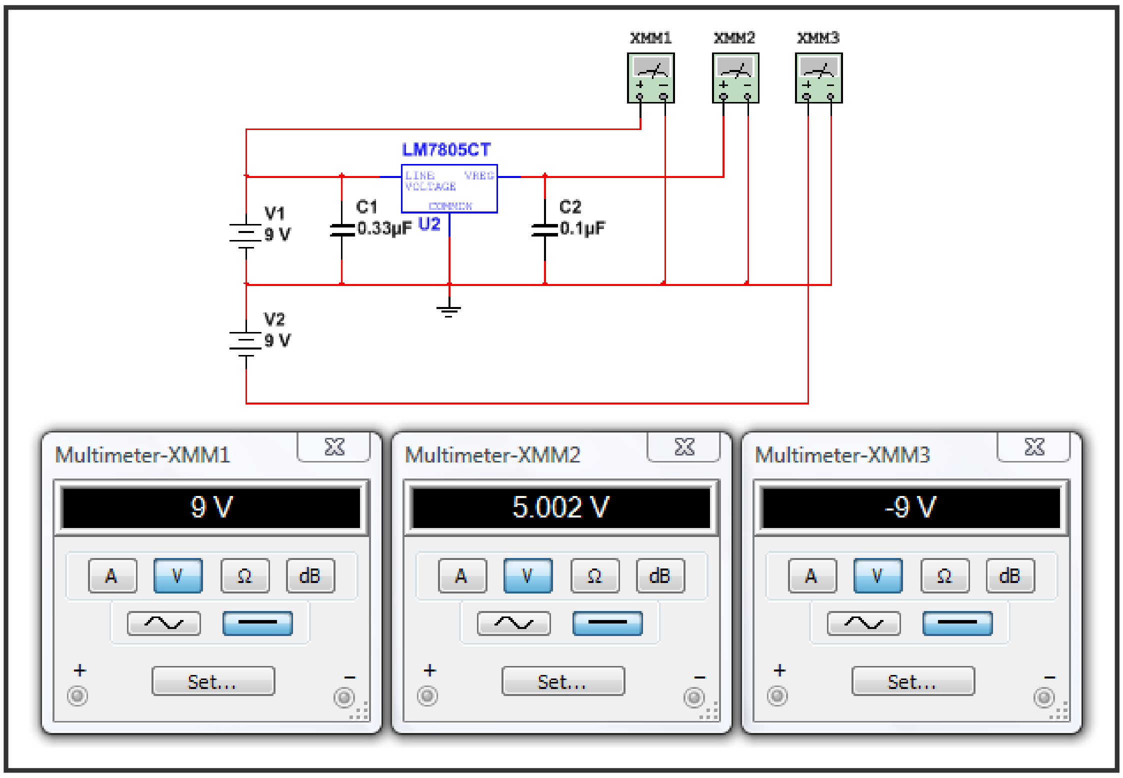
Change State:

○ Turn ON

○ Turn OFF

Submit

Appendix-8: Power Supply Connection.



Multimeter-XMM1

9 V

Multimeter-XMM2

5.002 V

Multimeter-XMM3

-9 V

Appendix-9: PIC18F2420 Code.

```
//==================================================================
//Project Title  : Security Alarm System
//Student Names  : Mohamed Alsubaie, Ahmed Alkhwher and Naif Alharbi
//Student ID NO.: 09562211, 10981738 and 09564323
//Supervisor     : Professor Nicholas Bowring
//==================================================================

//Include Headers Files
#include <p18f2420.h>
#include <delays.h>

//Microcontroller Setup
#pragma config WDT=OFF , OSC=INTIO7 , PWRT = ON, LVP=OFF, MCLRE = OFF

//LCD Commands
#define CLEAR_SCREEN     0b00000001
#define FOUR_BIT         0b00101100
#define LINES_5X7        0b00111000
#define CURSOR_OFF       0b00001101
#define CURSOR_BLINK     0b00001111
#define CURSOR_RIGHT     0b00000110

#define DATA_PORT        PORTA            //Connect the LCD data to PORTA
#define RS_PIN           PORTCbits.RC0    //RA6 is connected to RS pin on LCD
#define E_PIN            PORTCbits.RC1    //RA7 is connected to E pin on LCD

#define sensor           PORTCbits.RC3    //RA5 is connected to the sensor
#define LED              PORTBbits.RB7    //RB7 is connected to the buzzer

//Inputs From Arduino & Ethernet
#define serverPinON      PORTCbits.RC4    //ON button from the web page (Pin C4)
#define serverPinOFF     PORTAbits.RA4    //OFF button from the web page (Pin A4)

//Outputs To Arduino & Ethernet
#define statePin         PORTCbits.RC5    //(Pin C5) is used to provide the system state (ON/OFF) to Arduino
#define motionPin        PORTCbits.RC6    //(Pin C6) is used to provide the motion detection to Arduino (also
to LED)
#define voicePin         PORTCbits.RC7    //(Pin C7) is used to provide the voice detection to Arduino (also to
LED)

int voice;                                //integer to save the Microphone value from the ADC
int voiceThreshold = 400;                 //voice threshold used for detection condition
int voice_condition = 0;                  //voice condition
int motion_condition = 0;                 //motion condition

int thePassword[] = {1, 2, 3, 4};         //the default password is 1234, the password is stored in this array
(xxxx)
int time[];                               //the time values is stored in this array (00:00)
int numberPressed = 12;                   //because no numberPresswd is equal to 12
int key = 12;                             //because no key is equal to 12
int password = 0;                         //used to check the right password
int resetPassword = 0;                    //reset password condition
int x = 0;                                //x is used as a counter and for condition
int hour = 0;                             //hour integers values
int min = 0;                              //min integers values
int show = 0;                             //show condition
int state = 0;                            //state condition
int warning = 0;                          //warning condition

int LW = 0;                               //Last Warning condition
int LW_time[] = {0, 0, 0, 0};             //array to store the Last Warning time

//==================================================================
//Liquid Crystal Display Functions
//==================================================================
void SetAddr(unsigned char DDaddr)
{
        DATA_PORT &= 0xf0;               // Write upper nibble
        DATA_PORT |= (((DDaddr | 0b10000000)>>4) & 0x0f);

        RS_PIN = 0;                       //Set control bit
        Delay10TCYx(5);                   //20 clock cycles, 2.5mS
        E_PIN = 1;                      //Clock the cmd and address in
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 0;

        DATA_PORT &= 0xf0;              //Write lower nibble
        DATA_PORT |= (DDaddr&0x0f);

        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 1;                      //Clock the cmd and address in
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 0;
}

void WriteCmd(unsigned char cmd)
{
        DATA_PORT &= 0xf0;
        DATA_PORT |= (cmd>>4)&0x0f;
        RS_PIN = 0;                      //Set control signals for command
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 1;                       //Clock command in
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 0;

        // Lower nibble interface
        DATA_PORT &= 0xf0;
        DATA_PORT |= cmd&0x0f;
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 1;                       //Clock command in
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 0;
}

void WriteChar(char data)
{
        DATA_PORT &= 0xf0;
        DATA_PORT |= ((data>>4)&0x0f);

        RS_PIN = 1;                      //Set control bits
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 1;                       //Clock nibble into LCD
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 0;

        DATA_PORT &= 0xf0;               //Lower nibble interface
        DATA_PORT |= (data&0x0f);

        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 1;                       //Clock nibble into LCD
        Delay10TCYx(5);                  //20 clock cycles, 2.5mS
        E_PIN = 0;
}

void WriteString(const rom char *buffer)
{
        while(*buffer)                   //Write data to LCD up to null
        {
                Delay10TCYx(5);          //20 clock cycles, 2.5mS
                WriteChar( *buffer);     //Write character to LCD
                buffer++;                //Increment buffer
        }
        return;
}

//==================================================================
//Keypad Function
//==================================================================
void scan()
{
        //B4 is 0 scanning 1,4,7,*
        PORTBbits.RB4 = 0;
        PORTBbits.RB5 = 1;
        PORTBbits.RB6 = 1;

        if (PORTBbits.RB0==0)
```

```
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB0==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 1;
        return;
}

if (PORTBbits.RB1==0)
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB1==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 4;
        return;
}

if (PORTBbits.RB2==0)
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB2==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 7;
        return;
}

if (PORTBbits.RB3==0)
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB3==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 10;             //10 is for "ON" pressed
        return;
}

//B5 is 0 scanning 2,5,8,0
PORTBbits.RB4 = 1;
PORTBbits.RB5 = 0;
PORTBbits.RB6 = 1;

if (PORTBbits.RB0==0)
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB0==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 2;
        return;
}

if (PORTBbits.RB1==0)
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB1==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 5;
        return;
}

if (PORTBbits.RB2==0)
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB2==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 8;
        return;
}

if (PORTBbits.RB3==0)
{
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        while (PORTBbits.RB3==0);       //wait until switch has been released
        Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
        numberPressed = 0;
        return;
        }

        //B6 is 0 scanning 3,6,9,#
        PORTBbits.RB4 = 1;
        PORTBbits.RB5 = 1;
        PORTBbits.RB6 = 0;

        if (PORTBbits.RB0==0)
        {
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                while (PORTBbits.RB0==0);       //wait until switch has been released
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                numberPressed = 3;
                return;
        }

        if (PORTBbits.RB1==0)
        {
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                while (PORTBbits.RB1==0);       //wait until switch has been released
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                numberPressed = 6;
                return;
        }

        if (PORTBbits.RB2==0)
        {
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                while (PORTBbits.RB2==0);       //wait until switch has been released
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                numberPressed = 9;
                return;
        }

        if (PORTBbits.RB3==0)
        {
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                while (PORTBbits.RB3==0);       //wait until switch has been released
                Delay100TCYx(8);                //wait 0.1s for switch bounce to stop
                numberPressed = 11;             //11 is for "OFF" pressed
                return;
        }
}

//==================================================================
//Secyrity System Functions
//==================================================================
void TimeValues()                       //this function is storing the time values in time[]
{
        time[0] = hour/10;              //store the hour tens number in time[0]  dividing by 10
        time[1] = hour%10;              //store the hour units number in time[1] dividing by 10 but taking
the reminder
        time[2] = min/10;               //store the min tens number in time[2]    dividing by 10
        time[3] = min%10;               //store the min units number in time[3]   dividing by 10 but taking
the reminder
}

void ShowTime()                         //this function is displaying the time from time[]
{
        //write the time in this form(      hh:mm     ), hour and min are having 2 numbers,
        //they display by splitting them into tens and units number and add each of them to 0x30
        //as it shown in the LCD Character Set, for example 0x30 + 5 will display character 5
        SetAddr(0xC5);                          //move cursor to the middle
        WriteChar(0x30 + time[0]);              //display the tens number of hour that stored in time[0] as
character
        WriteChar(0x30 + time[1]);              //display the units number of hour that stored in time[1] as
charactar
        WriteChar(':');                         //display character :
        WriteChar(0x30 + time[2]);              //display the tens number of min that stored in time[2] as
charactar
        WriteChar(0x30 + time[3]);              //display the units number of min that stored in time[3] as
charactar
        WriteString("      ");                  //display a string of spaces, just for better visualization
}
```

```
void ShowLWTime()       //this function is displaying the Last Warning time from LW_time[]
{
        if(LW == 1)                             //if their Last warning display it
        {
                SetAddr(0xC5);                  //move cursor to the middle
                WriteChar(0x30 + LW_time[0]);   //display the tens number of hour that stored in LW_time[0]
as charactar
                WriteChar(0x30 + LW_time[1]);   //display the units number of hour that stored in LW_time[1]
as charactar
                WriteChar(':');                 //display character :
                WriteChar(0x30 + LW_time[2]);   //display the tens number of min that stored in LW_time[2] as
charactar
                WriteChar(0x30 + LW_time[3]);   //display the units number of min that stored in LW_time[3]
as charactar
                WriteString("      ");          //display a string of spaces, just for better visualization

        }

        else                                    //else display nothing
        {
                SetAddr (0xC0);
                WriteString("                ");
        }
}

void systemState()                              //this function displays the state of the system: ON, OFF or
Warning
{
        //write Security System in the first line
        SetAddr (0x80);
        WriteString("Security System ");
        //write the system state on the second line depending on the right condition
        SetAddr (0xC0);
        if(state == 1 && warning == 0){WriteString("State: ON       ");}
        else if(state == 1 && warning == 1){WriteString("State: Warning! ");}
        else if (state == 0){WriteString("State: OFF      ");}
}

void enterPassword()                            //this function is to get the password
{
        SetAddr(0xC0);                          //move the cursor to the beginning of the second line
        WriteString("Password: ");              //ask for the password
        SetAddr(0xCA);                          //move the cursor to the middle of the second line

        for(x = 0; x < 4; x++)                  //get four numbers for the password
        {
                 numberPressed = 12;            //there is no key with this value (12)
                 while(numberPressed > 9)       //disable ON and OFF keys while entering the password
                 {
                        scan();                 //call scan function
                        key = numberPressed;    //save the value of numberPressed in integer key
                 }                              //repeat that while nothing is pressed or ON/OFF key are
pressed

                 //when the key gets a number (0-9) write x on the display
                 //(the number that the user pressed for password is hidden, it will not shown in LCD, it will
represented by x)
                 WriteChar('x');

                 //check this number with the order of the password in thePassword[x], where x represent the
sequence of the right password
                 if(key == thePassword[x])      //if the value of key is the same as the sequence of numbers
stored in thePassword[x]
                 {
                        password++;             //increase integer password by 1
                 }
        }

        //do the right condition depending on password, state and resetPassword
        if(password == 4 && state == 0 && resetPassword == 0){state = 1;}        //if this condition is true
change state from 0 to 1
        else if(password == 4 && state == 1 && resetPassword == 0){state = 0;}   //if this condition is true
change state from 1 to 0
```

```
        else if(password == 4 && resetPassword == 1){resetPassword = 3;}         //if this condition is true
change resetPasword from 1 to 3

        //when all these condition are not true then show Wrong Password! on the display for about 2 second
and the disappear
        else
        {
                SetAddr(0xC0);
                WriteString("Wrong Password! ");
                Delay100TCYx(160);
                SetAddr(0xC0);
                WriteString("                ");
        }

        password = 0;   //change the value of password to 0 to make sure that it password will count
correctly for the next time
}

//==================================================================
//Interrupt Function
//==================================================================
#pragma code isr = 0x08                         //puts isr in memory location 0x08
#pragma interrupt isr                           //declares isr as my interrupt (service) routine

void isr(void)
{
        if (INTCONbits.TMR0IF)                  //Interrupt Check
        {
          INTCON = 0b10100000;                  //sets GIE,TMR0IE
      INTCONbits.TMR0IF = 0;                    //clear TMR0IF
      //set the right value for TMR0H and TMR0L to give a 60 second
          TMR0H = 0xF8;
          TMR0L = 0x7F;

          min++;                                //increase the min by 1
          if(min == 60){min = 0; hour++;}       //when min reach 60 replace it with 0 and increase the hour
by 1
          if(hour == 24){hour = 0;}             //when hour reach 24 replace it with 0

          TimeValues();                         //store the time value in time[0-3]

          //while interrupting each 1 min the time will increase and will display on the LCD if show is equal
to 1
          if(show == 1)
      {
                ShowTime();                     //show the updated time
          }
        }
}

#pragma code                                    // Return to the default code section

//==================================================================
//Main Function
//==================================================================
void main (void)
{
        //========SETUP========//
        OSCTUNEbits.INTSRC = 0;         //To select 31kHz CLK
        OSCCON = 0b00000000;            //OSCCON CLK to 31kHz

        ADCON1 = 0b11101111;            //all IO are digital, except AN4 (on RA5) to read the microphone
values
        TRISA = 0b00110000;             //sets PORTA: all as outputs (LCD), except RA5 (Mic) and RA4
(serverPinOFF) as inputs
        PORTA = 0b00000000;             //turns off PORTA outputs , good start position
        TRISB = 0b00001111;             //sets PORTB: RB0-RB3 as inputs (Keypad), and RB4-RB7 as outputs
(Keypad and LED)
        PORTB = 0b00000000;             //turns off PORTB outputs, good start position
        TRISC = 0b00011000;             //sets PORTC: all as outputs (Buzzer, statePin, motionPin and
voicePin), except RC3 (PIR) and RC4 (serverPinON) as inputs
        PORTC = 0b00000000;             //turns off PORTC outputs, good start position
        T0CON = 0b10000111;             //sets TMR0 on, 16bits, internal clock, prescaler assigned, prescaler
= 256
        INTCON2bits.RBPU=0;             //turn on the pull-up resistors (in PORTB)

        ADCON0bits.CHS2 = 1;            //select the right channel for ADC
        ADCON0bits.ADON = 1;            //turn on A/D
        ADCON2 = 0b10000000;            //right justified

        //LCD configuration
        WriteCmd ( 0x02 );                      //sets 4bit operation
        WriteCmd ( FOUR_BIT & LINES_5X7 );      //sets 5x7 font and multiline operation.
        WriteCmd ( CURSOR_BLINK );              //blinks cursor
        WriteCmd ( CURSOR_RIGHT  );             //moves cursor right

        //show a welcome message on the screen
        SetAddr(0x80);                          //set the cursor at the beginning of the first line
        WriteString("Welcome To The  ");        //show the welcome massage
        SetAddr(0xC0);                          //set the cursor at the beginning of the second line
        WriteString("Security System ");        //show this line on the display
        Delay100TCYx(195);                      //delay about 2.5 sec
        WriteCmd (CLEAR_SCREEN);                //clear the screen

        x = 0;                  //make sure that x = 0 to enter the while loop
        while (x < 1)           //the condition of the while loop is when x is less than 1
        {
                //display tha following on the LCD
                SetAddr (0x80);
                WriteString("Enter The Time  ");
                SetAddr (0xC0);
                WriteString("     00:00      ");

                for(x = 0; x < 4; x++)          //get four numbers for the time
                {
                         if(x == 0) {SetAddr (0xC5);}           //start by placing the cursor at the first 0
of the time
                         else if(x == 2) {WriteChar(':');}      //before entering the 3 number re-write : and

then continue

                         numberPressed = 12;     //there is no key with this value (12)
                         while(numberPressed > 9)               //disable ON and OFF keys while entering the
time
                         {
                            scan();                             //call the scan function
                                time[x] = numberPressed;        //store the value of numberPressed in time[x]
                         }
        //repeat that while nothing is pressed or ON/OFF key are pressed

                         WriteChar(0x30 + time[x]); //display the number pressed on the LCD
                }
                WriteString("      ");          //write a few spaces to get-rid of the cursor
                Delay100TCYx(80);               //delay about 1 sec

                hour = time[1] + (time[0] * 10); //calculate the hour value from the time[0-1]
                min = time[3] + (time[2] * 10);  //calculate the min value from the time[2-3]

                //note: after all that x is equal to 3 and it will not do this again
                //check if the time is valid or invalid
                //if the time entered is an invalid time [ask for enter a valid time]
                if(hour > 23 || min > 59)
                {
                        SetAddr (0x80);
                        WriteString("!!Please Enter!!");
                        SetAddr (0xC0);
                        WriteString("!!a valid time!!");
                        Delay100TCYx(240);              //delay about 3 second
                        WriteCmd (CLEAR_SCREEN);        //clear the screen
                        x = 0;                          //in this case x is change from 3 to 0 to re-enter
the valid time
                }
        }

        INTCON = 0b10100000;            //sets GIE,TMR0IE
        INTCONbits.TMR0IF = 0;          //clear TMR0IF
        //set the right value for TMR0H and TMR0L, (0xFFFF - 0xF87F = 0x0780, 0x0780 = 1920d = 32 x 60 = 60S)
        TMR0H = 0xF8;
        TMR0L = 0x7F;

        TimeValues();                   //call this function to store the time entered values
        systemState();                  //show the system state of the LCD

        while (1)
        {
                numberPressed = 12;             //there is no key with this value (12)
                while(numberPressed > 11)       //enable ON and OFF keys
                {
                        ADCON0bits.GO_DONE = 1;           // do A/D measurement
                        while (ADCON0bits.GO_DONE == 1);   //wait until bit = 0  measurement completed
                        voice = ADRESL + (ADRESH * 256);  //store the value in integer voice

                        scan();                 //scaning the kepad
                        key = numberPressed;    //store the value of the kepad in integer key

                        //Inputs From Arduino & Ethernet
                        if((serverPinON == 1)&&(state == 0)){state = 1; systemState();}          //only if 
the system is off, turn it on
                        if((serverPinOFF == 1)&&(state == 1)){state = 0; systemState();}  //only if the system 
is on, turn it off

                        //voice detection condition, if true voicePin (and LED) = 1 to update the web page
                        if((voice > voiceThreshold)&&(state == 1)&&(voice_condition == 0)){voicePin = 1;
voice_condition = 1;}

                        //motion detection condition, if true motionPin (and LED) = 1 to update the web page
                        if((sensor == 1)&&(state == 1)&&(motion_condition == 0)){motionPin = 1;
motion_condition = 1;}

                        //when the systemState is on and this condition is true turn on the buzzer and the 
warning state
                        if((state == 1) && (sensor == 1 || voice > voiceThreshold) && (LED == 0))
                        {
                                warning = 1;            //turn on warning condition
                                LED = 1;                //turn on buzzer
                                statePin = 1;           //turn on statePin to update the state on the web 
page

                                //PWM 50% 500Hz
                                T2CON = 0b00000100;     //prescaler + turn on TMR2;
                                PR2 = 0b00001111;       //set the right value for PR2
                                CCPR1L = 0b00000111;    //set duty MSB
                                CCP1CON = 0b00111100;   //duty lowest bits + PWM mode

                                WriteCmd (CLEAR_SCREEN);        //clear the screen

                                //display warning and the time that the state has started
                                SetAddr (0x80);
                                WriteString(" !!!Warning!!!  ");
                                ShowTime();
                                //store the warning time on LW array
                                LW = 1;
                                LW_time[0] = time[0];
                                LW_time[1] = time[1];
                                LW_time[2] = time[2];
                                LW_time[3] = time[3];
                        }

                        //when the systemState is turned off the buzzer will turned off as will as the 
warning state and other required signals
                        else if (state == 0)
                        {
                                warning = 0;
                                LED = 0;
                                voice_condition = 0;
                                voicePin = 0;
                                motion_condition = 0;
                                motionPin = 0;
                                statePin = 0;

                                //PWM 0%
                                T2CON = 0b00000100;      //prescaler + turn on TMR2;
                                PR2 = 0b00000111;        //set the right value for PR2
                                CCPR1L = 0b00000000;     //set duty MSB
                                CCP1CON = 0b00001100;    //duty lowest bits + PWM mode
                        }

                }

                switch(key)             //use switch statement for the integer key, which is the keypad value
            {
                   //===============System State===============//
                case 1:                          //case 1 is to show the system state
                {
                           show = 0;             //show is 0 to make sure that the updated time will not
display while intruping each 1 min
                           systemState();        //show the system state
                           break;
                }

                   //===============Show The Time===============//
                   case 2:                       //case 2 is to show the time
                {
                           show = 1;             //show is 1 to make sure that the updated time will display
while intruping each 1 min
                           //clear the screen and display the time
                           WriteCmd (CLEAR_SCREEN);
                           SetAddr (0x80);
                           WriteString("    The Time    ");
                           ShowTime();
                           break;
                }

                   //===============Reset Password===============//
                   case 3:                       //case 3 is to reset password (or change password)
           {
                           show = 0;             //show is 0 to make sure that the updated time will not
display while interrupting each 1 min

                           //clear the screen and display reset password for about 1.5 second
                           WriteCmd (CLEAR_SCREEN);
                           SetAddr(0x80);
                           WriteString("Reset Password  ");
                           Delay100TCYx(120);

                           //set the cursor back to the beginning of the first line and write the following
                           SetAddr(0x80);
                           WriteString("Enter The Old   ");

                           resetPassword = 1;            //change the condition of resetPassword to 1 to make
sure that it do the right procedure of reset password
                           enterPassword();              //call this function to deal with the password
procedure
                           WriteCmd (CLEAR_SCREEN);      //clear the screen

                           //in the enterPassword function:
                           //when the password is wrong display a wrong password message and keep
resetPassword 1, and go back to system state
                           //when the password is correct resetPassword will change to 3, and then the user
will asked to enter the new password
                           if(resetPassword == 3)
                           {
                                //clear screen and ask for the new password
                                WriteCmd (CLEAR_SCREEN);
                                SetAddr (0x80);
                                WriteString("Enter The New   ");
                                SetAddr(0xC0);
                                WriteString("Password: ");
                                SetAddr(0xCA);

                                //take four numbers of the new password
                                for(x = 0; x < 4; x++)
                                {
                                  numberPressed = 12;                              //there is no key
with this value (12)
```

```
                                  while(numberPressed > 9)                                //disable ON and OFF
keys while entring the new password
                                  {
                                        scan();
        //scan the keypad
                                        thePassword[x] = numberPressed;  //write the new password over the
old password in thePassword[0-3] (overwrite)
                                  }
        //repeat that while nothing is pressed or ON/OFF key are pressed

                                  WriteChar(0x30 + thePassword[x]);      //display the number that been
pressed on the LCD
                                }
                                WriteString("  ");                                              //write a
few spaces to get-rid of the cursor
                                Delay100TCYx(80);                                               //delay
about 1 second
                                SetAddr(0xC0);                                                  //move
cursor to the beginning of the second line
                                WriteString("Password: xxxx  ");         //convert the password to xxxx
                                Delay100TCYx(80);                                               //delay
about 1 second
                          }
                          resetPassword = 0;                                            //change
resetPassword back to 0
                          systemState();                                                //show the system
state
                          break;
            }

                    //===========Show Last Warning Time===========//
                        case 4:
            {   //clear the screen and show the last warning time
                                show = 0;                       //show is 0 to make sure that the updated
time will not display while intruping each 1 min
                                WriteCmd (CLEAR_SCREEN);
                                SetAddr (0x80);
                                WriteString("Last Warning    ");
                                ShowLWTime();
                                break;
                        }

                        //=================ON State==================//
                        case 10:
            {
                                //when system state is already ON disable this key by doing nothing
                                if(state == 0)          //when the system state is OFF enable ON key
                                {
                                        show = 0;
        //show is 0 to make sure that the updated time will not display while intruping each 1 min
                                        WriteCmd (CLEAR_SCREEN);         //clear the screen
                                        SetAddr (0x80);                                         //move
cursor to the beginning of the first line
                                        WriteString("Security System "); //display this message

                                        //in the enterPassword function, the programme will ask for the
password, if it correct to turn the system,
                                        //if it wrong = wrong password message and keep the system state
without changing it
                                        enterPassword();
                                        systemState();          //display the system state

                                }
                                break;
            }

                        //=================OFF State=================//
                        case 11:
            {
                                //when system state is already OFF disable this key by doing nothing
                                if(state == 1)          //when the system state is ON enable OFF key
                                {
                                        show = 0;
        //show is 0 to make sure that the updated time will not display while interrupting each 1 min
```

```
                                        WriteCmd (CLEAR_SCREEN);        //clear the screen
                                        SetAddr (0x80);                                         //move
cursor to the beginning of the first line
                                        WriteString("Security System "); //display this message

                                        //in the enterPassword function, the programme will ask for the
password, if it correct to turn the system,
                                        //if it wrong = wrong password message and keep the system state
without changing it
                                        enterPassword();
                                        systemState();                  //display the system state

                                }
                                break;
            }
                    }
        }
}
```

Appendix-10: Arduino Code.

```
//================================================================
//Project Title  : Security Alarm System
//Student Names  : Mohamed Alsubaie, Ahmed Alkhwher and Naif Alharbi
//Student ID NO.: 09562211, 10981738 and 09564323
//Supervisor     : Professor Nicholas Bowring
//================================================================

//Include Headers Files (libraries)
#include <Ethernet.h>
#include <SPI.h>

//Outputs to the PIC Microcontroller
#define  offPin    19    //the responsible pin to turn off the system
#define  onPin     18    //the responsible pin to turn on the system

//Inputs From the PIC Microcontroller
#define  statePin  17    //system state
#define  PIRPin    16    //PIR (movement detection)
#define  MicPin    15    //Microphone (voice detection)

int state;      //integer to hold the state in the Arduino Microcontroller
int statePIC;    //integer to read the PIC state
int PIR;         //integer to read the PIR state
int Mic;         //integer to read the Mic state

//Mac Address of the provided Ethernet Shield
byte mac[] = { 0x90, 0xA2, 0xAD, 0x00, 0x5B, 0x16 };

EthernetServer server(80);

void setup()
{
        Serial.begin(9600);                  //initialize the serial communications
        Ethernet.begin(mac);                 //set up the Ethernet
        server.begin();
        Serial.print("My IP Address: ");     //print the IP address on the serial monitor
        Serial.println(Ethernet.localIP());

        //set the inputs of the Arduino
        pinMode(statepin, INPUT);
        pinMode(PIRPin, INPUT);
        pinMode(MicPin, INPUT);

        //set the outputs of the Arduino
        pinMode(onPin, OUTPUT);
        pinMode(offPin, OUTPUT);

        delay(500);                      //delay for 0.5s
        state = digitalRead(statePin);   //Hold the current state
}

void loop()
{
      EthernetClient client = server.available();

      if (client)
      {
        //an http request ends with a blank line
        boolean currentLineIsBlank = true;
        String buffer = "";

        //Declare the buffer variable
        while (client.connected())
        {
          if (client.available())
          {
            char c = client.read();
            Serial.print(c);  //Send every character read to serial port
            buffer+=c;        //Assign to the buffer

            if (c == '\n' && currentLineIsBlank)
            {
              //send a standard http response header
              client.println("HTTP/1.1 200 OK");           //Standard HTTP response
              client.println("Content-Type: text/html\n");
              client.println("<meta http-equiv=\"refresh\"content=\"5");

              //return the status value to 0, in order to prevent from crashes with the PIC microcontroller
              //in other word if status value stay 2 the user will not be able to turn off the system from the
keypad
              //the same with status = 1. so it necessarily to return this value to 0 which will give a
trigger signal
              //that can be detected from the PIC microcontroller.
              if(state < 2){client.print("; URL=http://"); client.print(Ethernet.localIP());
client.print("/?status=2\">");}
              else {client.println("\">");}

              //show the MMU logo and the Project Title
              client.println("<body>");
              client.println("<img src =
\"http://www2.docm.mmu.ac.uk/STAFF/N.Rattenbury/PopMathsQuiz/MMU_logo.jpg\" alt=\"Smiley face\"
style=\"float:right\" width=\"80\" height=\"100\"><h1>Security System </h1><hr>");

              //read the PIC Inputs
              statePIC = digitalRead(statePin);
              Mic = digitalRead(MicPin);
              PIR = digitalRead(PIRPin);

              //check the current state if (ON, OFF, of Warning!)
              client.println("<h3>Current State: ");
              if(statePIC == HIGH && PIR == LOW && Mic == LOW)
              {client.println("ON</h3>");}
              else if (statePIC == HIGH && (PIR == HIGH || Mic == HIGH))
              {client.println("Warning!</h3>");}
              else{client.println("OFF</h3>");}

              //check if PIR detact somthing
              client.println("<h3>Movement Detection: ");
              if(PIR == HIGH)
              {client.println("YES</h3>");}
              else{client.println("NO</h3>");}

              //check if Microphone detact somthing
              client.println("<h3>Voice Detection: ");
              if(Mic == HIGH)
              {client.println("YES</h3>");}
              else{client.println("NO</h3>");}

              //creat a radio buttons and submit button for user
              client.println("<h3>Change State: </h3>");  //show a headline
              client.print("<FORM action=\"http://");     //form an action
              client.print(Ethernet.localIP());           //print the IP address, which is the website name
              client.print("/\" >");
              //creat the first radio button and call it ON with status value of 2
              client.print("<P><INPUT type=\"radio\" name=\"status\" value=\"2\">Turn ON");
              //creat the second radio button and call it OFF with status value of 1
              client.print("<P><INPUT type=\"radio\" name=\"status\" value=\"1\">Turn OFF");
              //creat the submit button
              client.print("<P><INPUT type=\"submit\" value=\"Submit\"> </FORM>");
              client.println("</body>");
              break;
            }

        if (c == '\n')
        {
          // you're starting a new line
          currentLineIsBlank = true;
          buffer="";       //  Clear the buffer at end of line
        }

        else if (c == '\r')
        {
          //turn on the system through the internet if ON button is selected (trigger only)
          if(buffer.indexOf("GET /?status=1")>=0) { state = 2; digitalWrite(onPin, HIGH);}

          //turn off the system through internet if OFF button is selected (trigger only)
          if(buffer.indexOf("GET /?status=0")>=0) { state = 1; digitalWrite(offPin, HIGH);}

          //turn off both offPin and onPin after providing a trigger signal to the PIC Microcontroller
          if(buffer.indexOf("GET /?status=2")>=0) { state = 0; digitalWrite(onPin, LOW); digitalWrite(offPin, 
LOW);}
        }

        else
        {
          // you've gotten a character on the current line
          currentLineIsBlank = false;
        }
      }
    }

    delay(1);
    // close the connection:
    client.stop();
  }
}
```