TASK SCHEDULER USING ARM CONTROLLER (LPC1768)

Raghunathreddy Jangam

Final Project Report

ECEN 5613 Embedded System Design

December 12, 2015

Contents

CHAPTER 1
INTRODUCTION

1.1 Task Scheduler

Task Scheduler is a dedicated embedded device, which will remind you of the task to be performed by flashing RGB background color on LCD screen along with the buzzer sound. Task Scheduler has a keypad interface for user to set his/her tasks for respective time and date. To make the device user friendly, the device is installed with a resistive touch screen.

1.2 Objective of the work

The main objective of this project is to understand ARM architecture by using one of the ARM controllers. Other objectives of the system are to learn to interface new input and output modules.

1.3 System Overview

Sometimes during class, we tend to note down tasks to be done such as home works, assignments and quizzes. As most of the professors recommended students to not use mobile phones during the lectures, this Task Manager can be used to note down tasks on the spot.

As shown in Figure 1.1 this system uses ARM controller to control the inputs from the user through the hex keypad, touch screen and displays it on the output (LCD).

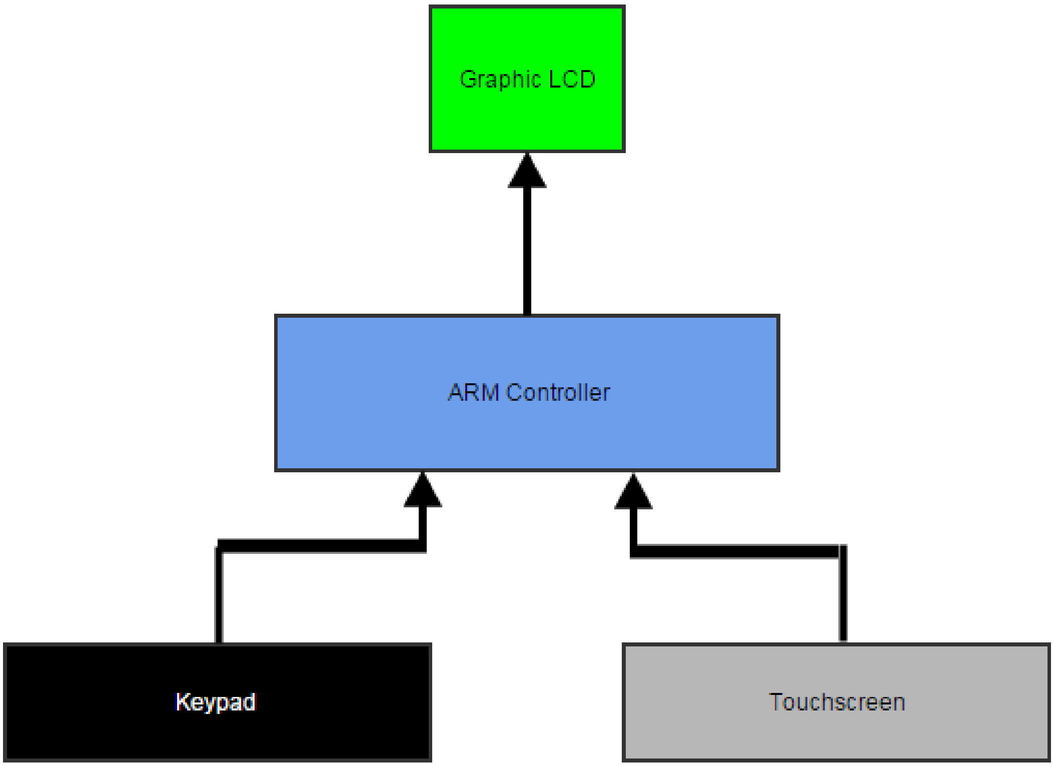


Figure 1.1 - System concept

1.4 Project work Schedule

November 2nd Week	➢ Finalized the project components ➢ Ordered the required parts
November 3rd Week	➢ Programmed LPC1768 ➢ Learnt to use uart for debugging
November 4th Week	➢ Interfaced keypad and wrote library for it ➢ Interfaced graphic LCD and wrote library for it ➢ Interfaced touch screen
December 1st Week	➢ Interfaced all the components and wrote the main code using these libraries ➢ Testing ➢ Put everything down on to a Perf board ➢ Started documentation

CHAPTER 2

TECHNICAL DESCRIPTION

The current chapter contains the exact details of the implementation of the Task Scheduler using ARM controller. This chapter contains Board Design followed by Firmware Design.

2.1 Board Design

2.1.1 Mbed Complier

The mbed compiler by ARM Corporation allows quick cloud based programming and easy uploading to the Processor. It features online libraries and debugging options as well as a code repository. This allows for fast programming as well as interfacing. The compiler allows programming in C++, compared to most other compilers like Arduino or Kiel, which only support C/embedded C based programming. The core platform and supporting libraries are developed under the Apache License 2.0, which allows the user to make modified versions of the software for distribution.

2.1.1 NXP LPC1768

The ARM processor used, NXP LPC1768, is a 32 bit RISC based ARM Cortex M3 processor, featuring 512KB of Flash and 32KB Ram, running at 96 MHz. It supports I2C, DAC, RTC, USB host device and I/O Interfaces. In this application, USB, RTC, repetitive interrupts and RTC functionalities are used. The processor is powered by USB input from the computer and also by a dedicated battery.

The mbed NXP LPC1768 Microcontroller in particular is designed for prototyping all sorts of devices, especially those including Ethernet, USB, and the flexibility of lots of peripheral interfaces and FLASH memory. It is packaged as a small DIP form-factor for prototyping with through-hole PCBs, strip board and breadboard, and includes a built-in USB FLASH programmer.

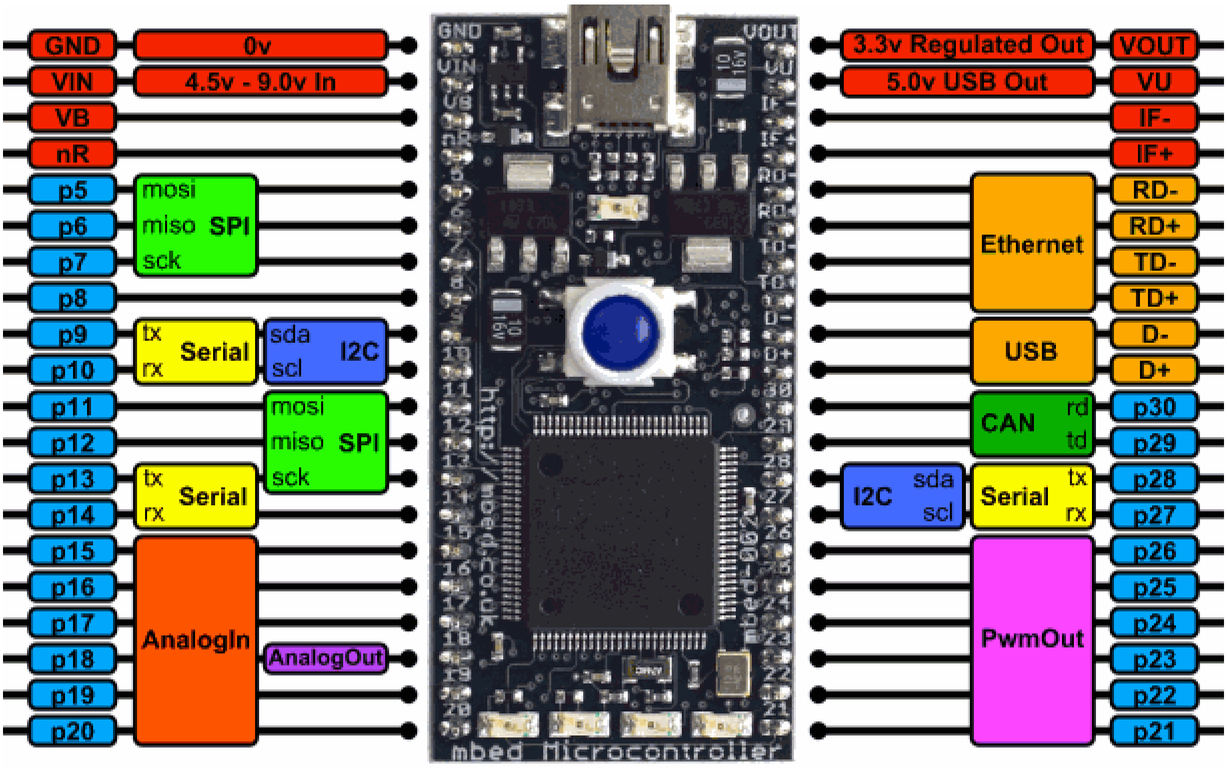


Figure 2.1 – Pin outs of mbed deveoplement board for LPC1768 (source: mbed boards)

2.1.3 Interfacing Keypad

The particular keypad shown in figure 2.2 by adafruit which also phas numeric and alphabets on the keypad ,this was driving factor to choose this particular part over the others. As you can see the backside of the keypad in figure 2.3 we can find 8 pins. The pins, starting from the left, are assigned column1, column2, column3, row1, row2, row3, row4 and the last one is not connected.

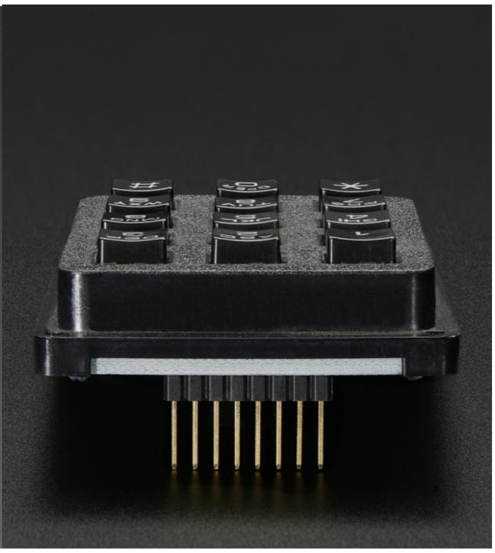


Figure 2.2 – Keypad-1 **Figure 2.3-Keypad2 (source:Adafruit)**

Continuity test is done on this to verify which pins are columns and rows as no information is given on the datasheet regarding this. A resistance of 200ohms was observed when a key was pressed between two particular pins. This interface requires only GPIO's of LPC1768. To use the alphabets there are 4 modes of the keypad and to indicate the mode in which the keypad is currently in, four LEDs are interfaced to the controller.

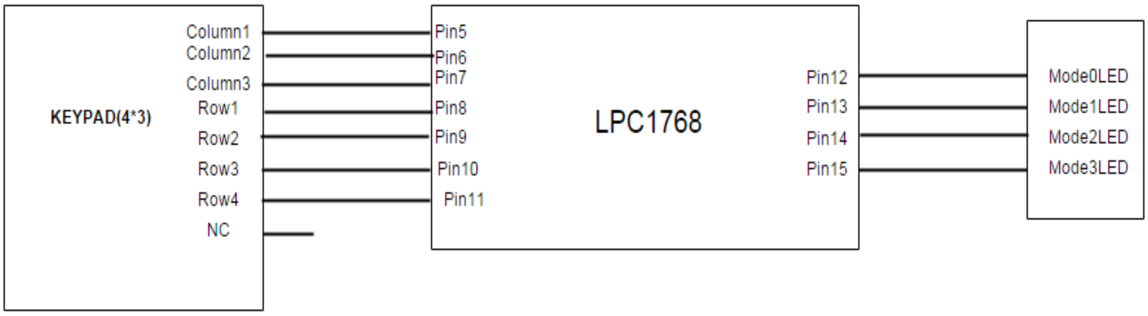


Figure 2.4 – Connection diagram of controller to keypad and LED's

As show in figure 2.4, pins are used for the Keypad alone and 4 pins for indicating mode of the keypad.

2.1.4 Interfacing Graphic LCD

Figure 2.5 is a LCD by Nintendo distributed by adafruit. As you can see in the figure 2.6 and 2.7, the pin names indicated, SID and SCLK, may lead to the misconception that communication is through I2C. When the datasheet is observed carefully, it mentions serial communication and also mentions the exact protocol used which is shown in the figure. To achieve the protocol, the bit banging is done using the pins which are connected to GPIO as show in figure 2.7. The power is supplied through the USB which is later dropped down to 3.3V with the help of an internal voltage regulator provided by mbed development board.

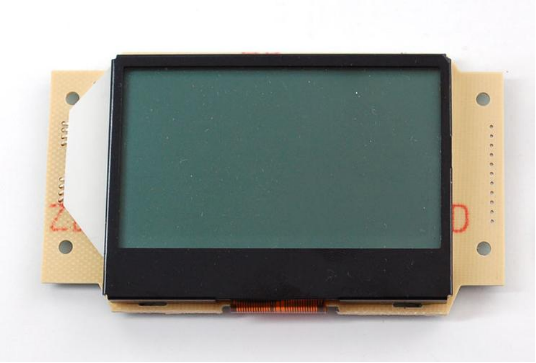


Figure 2.5 –Graphic LCD (source:Adafruit)

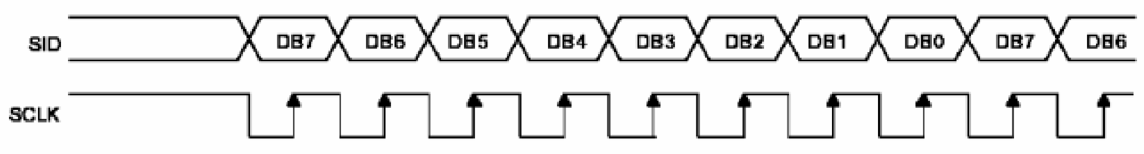


Figure 2.6 –Communication protocol used by LCD (source:www.adafruit.com)

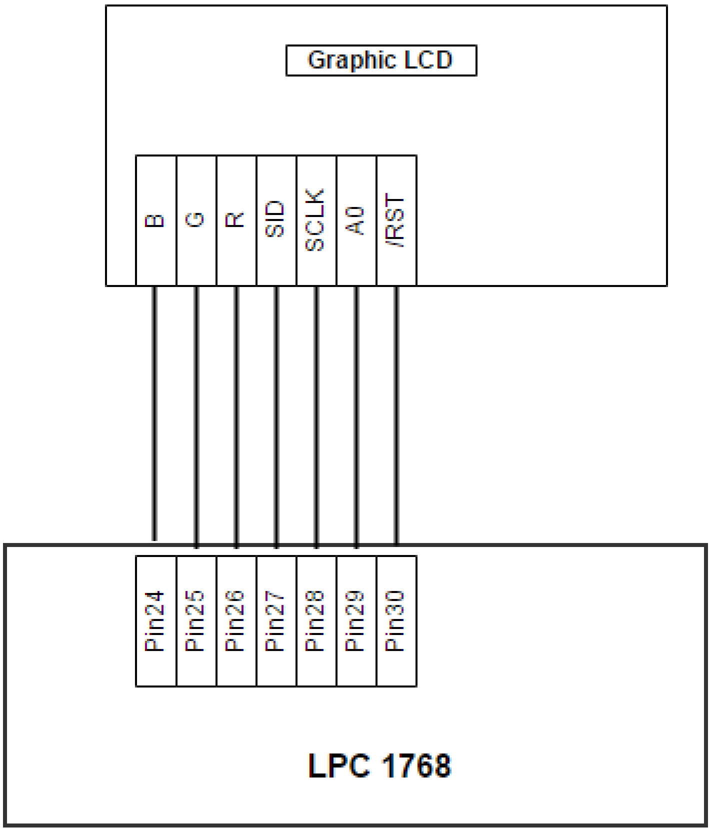


Figure 2.7 – Connection diagram of Graphic LCD with LPC1768

2.1.5 Interfacing Touch screen

Figure 2.8 shows the 4 wire resistive touch screen used. In this, one of the planes is energized with a 3.3VDC and the other plane gives an analog value between 0 and 3.3V. This indicates the position of the energized plane and the other plane position is determined by the same process. Figure 2.9 is a description of how it is connected to LPC1768. The value from the ADC was floating when touch screen is not pressed. Therefore, the two pins of the resistive touch screen are pulled down so that when it is in ideal condition, zero value is obtained from ADC block.

Figure 2.8 – resistive touchscreen (source :www.sparkfun)

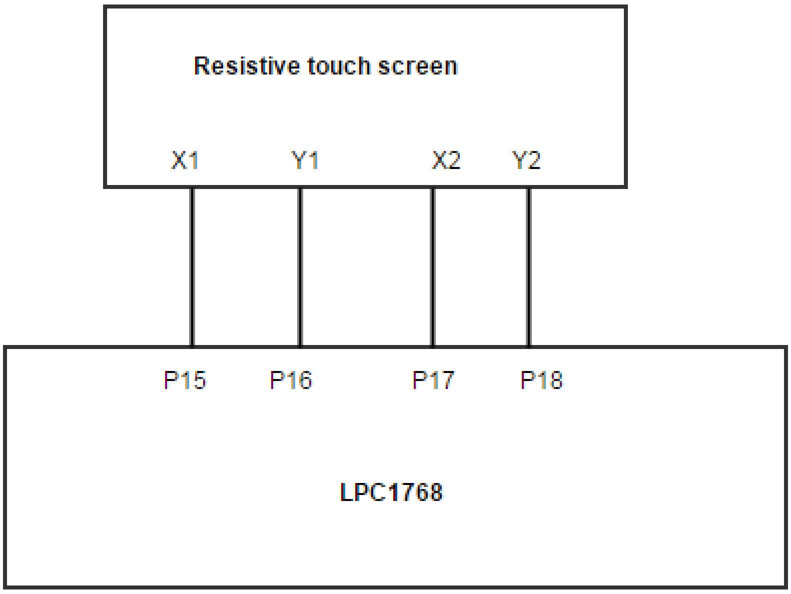


Figure 2.9 –Connection diagram of touch screen with LPC1768

2.2 Firmware Design

2.2.1 mbed complier

The mbed compiler by ARM Corporation allows quick cloud based programming and easy uploading to the processor. It features online libraries and debugging options as well as a code repository. This allows fast programming as well as interfacing. The compiler also allows programming in C++, as compared to most other compilers like Arduino or Kiel, which only support C/embedded C based programming. The core platform and supporting libraries are developed under the Apache License 2.0, which allows the user to make modified versions of the software for distribution.

2.2.2 Keypad driver

Requirements of the keypad is to use the keys not only for numeric characters but also alphabets and few special functions like backspace and enter into the next line. This requirement is fulfilled by making two keys dedicated for changing modes. Table 2-1, Table 2-2, Table 2-3, and Table 2-4 indicate the keys in different modes.

1	2	3
4	5	6
7	8	9
*	0	#

Table2-1.Mode – 0

Back Space	A	D
G	J	M
P	T	W
*	Q	#

Table2-2 Mode – 1

Enter	B	E
H	K	N
R	U	X
*	Z	#

Table2-3 Mode – 2

1	C	F
I	L	O
S	V	Y
*	Space	#

Table 2-4 Mode – 3

The following algorithm is used in this driver

Algorithm:

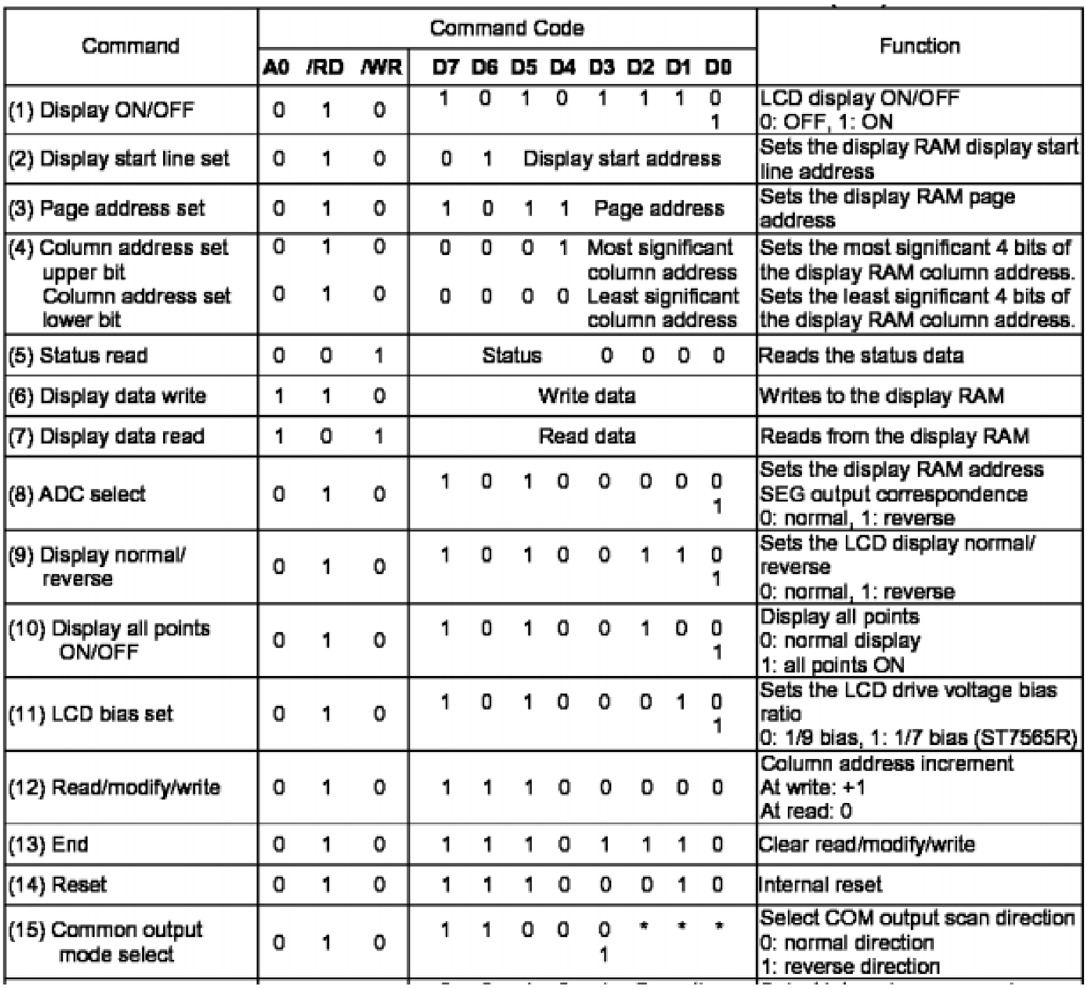
1. Initially, make all the input pins low
2. Check if there is an input in any of them. If yes, than implement the software debounce which is checks input after a certain delay. If no, than go back and wait for the input.
3. Now check if the value 1 is still present. If yes,
    a. Make column 1 zero and again check the value at row 1. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
    b. Make column 2 zero and again check the value at row 1. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
    c. Make column 3 zero and again check the value at row 1. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
4. Now check if the value 2 is still present. If yes,
    a. Make column 1 zero and again check the value at row 2. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
    b. Make column 2 zero and again check the value at row 2. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.

    c. Make column 3 zero and again check the value at row 2. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.

5. Now check if the value 3 is still present. If yes,
    a. Make column 1 zero and again check the value at row 3. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
    b. Make column 2 zero and again check the value at row 3. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
    c. Make column 3 zero and again check the value at row 3. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
6. Now check if the value 4 is still present. If yes,
    a. Make column 1 zero and again check the value at row 4. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
    b. Make column 2 zero and again check the value at row 4. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
    c. Make column 3 zero and again check the value at row 4. If the value still exists, we can update the value in the global variable and change the keypress count with an initial inspection of which mode it is in and next is step 7.
7. Led indication of exiting loop

2.2.2 LCD driver

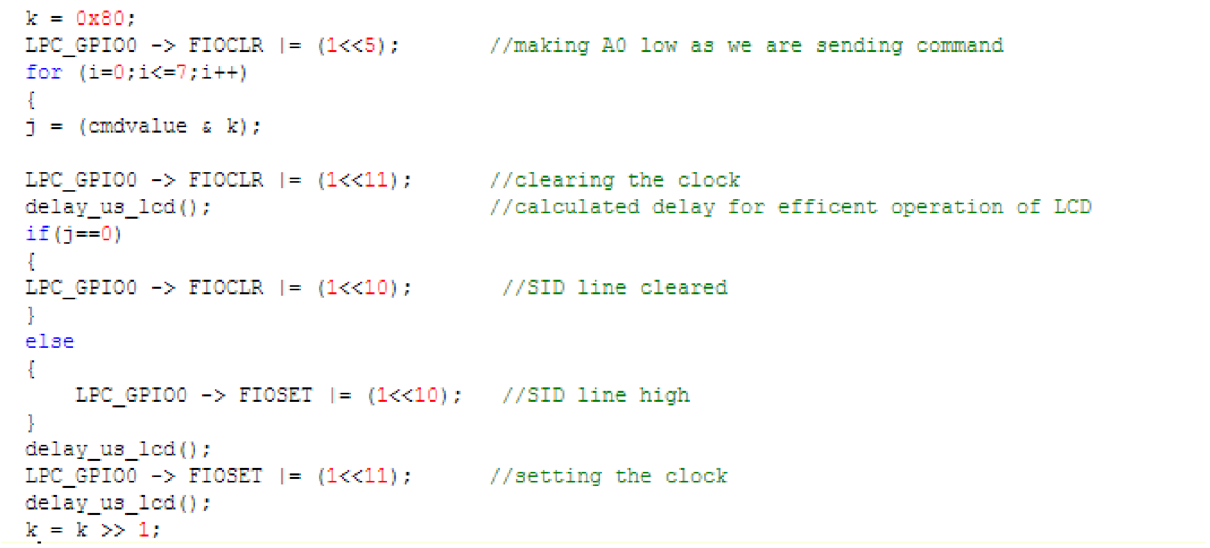
This particular LCD cannot be read, so the position of the pointer DDRAM must be closely monitored. It also lacks the capability of displaying cursor which again is developed through firmware. Figure 2-10 taken from the datasheet gives the information about various commands used by the LCD.



Command	Command Code											Function
	A0	/RD	/WR	D7	D6	D5	D4	D3	D2	D1	D0	
(1) Display ON/OFF	0	1	0	1	0	1	0	1	1	1	0 1	LCD display ON/OFF 0: OFF, 1: ON
(2) Display start line set	0	1	0	0	1	Display start address						Sets the display RAM display start line address
(3) Page address set	0	1	0	1	0	1	1	Page address				Sets the display RAM page address
(4) Column address set upper bit	0	1	0	0	0	0	1	Most significant column address				Sets the most significant 4 bits of the display RAM column address.
Column address set lower bit	0	1	0	0	0	0	0	Least significant column address				Sets the least significant 4 bits of the display RAM column address.
(5) Status read	0	0	1	Status				0	0	0	0	Reads the status data
(6) Display data write	1	1	0	Write data								Writes to the display RAM
(7) Display data read	1	0	1	Read data								Reads from the display RAM
(8) ADC select	0	1	0	1	0	1	0	0	0	0	0 1	Sets the display RAM address SEG output correspondence 0: normal, 1: reverse
(9) Display normal/ reverse	0	1	0	1	0	1	0	0	1	1	0 1	Sets the LCD display normal/ reverse 0: normal, 1: reverse
(10) Display all points ON/OFF	0	1	0	1	0	1	0	0	1	0	0 1	Display all points 0: normal display 1: all points ON
(11) LCD bias set	0	1	0	1	0	1	0	0	0	1	0 1	Sets the LCD drive voltage bias ratio 0: 1/9 bias, 1: 1/7 bias (ST7565R)
(12) Read/modify/write	0	1	0	1	1	1	0	0	0	0	0	Column address increment At write: +1 At read: 0
(13) End	0	1	0	1	1	1	0	1	1	1	0	Clear read/modify/write
(14) Reset	0	1	0	1	1	1	0	0	0	1	0	Internal reset
(15) Common output mode select	0	1	0	1	1	0	0	0 1	*	*	*	Select COM output scan direction 0: normal direction 1: reverse direction

Figure 2.10-LCD command codes

It has a serial communication but neither SPI or I2C which makes it necessary to bit banging. Figure 2-11 is the snap shot of the bit banging done to send the command instructions

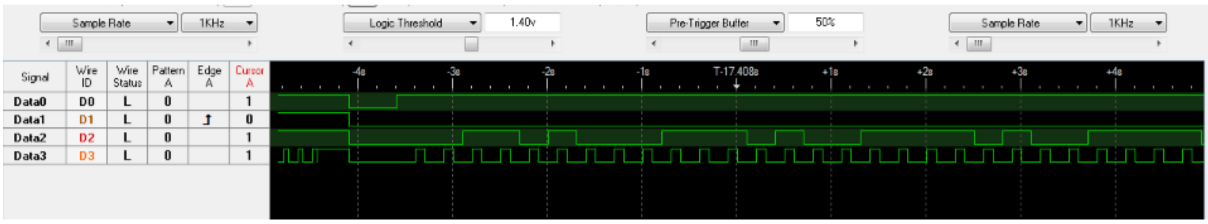


```
k = 0x80;
LPC_GPIO0 -> FIOCLR |= (1<<5);       //making A0 low as we are sending command
for (i=0;i<=7;i++)
{
j = (cmdvalue & k);

LPC_GPIO0 -> FIOCLR |= (1<<11);      //clearing the clock
delay_us_lcd();                      //calculated delay for efficent operation of LCD
if(j==0)
{
LPC_GPIO0 -> FIOCLR |= (1<<10);       //SID line cleared
}
else
{
    LPC_GPIO0 -> FIOSET |= (1<<10);   //SID line high
}
delay_us_lcd();
LPC_GPIO0 -> FIOSET |= (1<<11);      //setting the clock
delay_us_lcd();
k = k >> 1;
```

Figure 2.11-LCD bit banging

Figure 2-12 shows timing diagram of the bit banging done to communicate with LCD Data1 is A0, Data 2 is SID, Data 3 is SCLK



Sample Rate 1KHz | Logic Threshold 1.40v | Pre-Trigger Buffer 50% | Sample Rate 1KHz

Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Data0	D0	L	0		1
Data1	D1	L	0	↥	0
Data2	D2	L	0		1
Data3	D3	L	0		1

Figure 2.12- Bit Banging timing diagram

This LCD is not powered on if it does not go through the initialization sequence,which is mentioned in figure

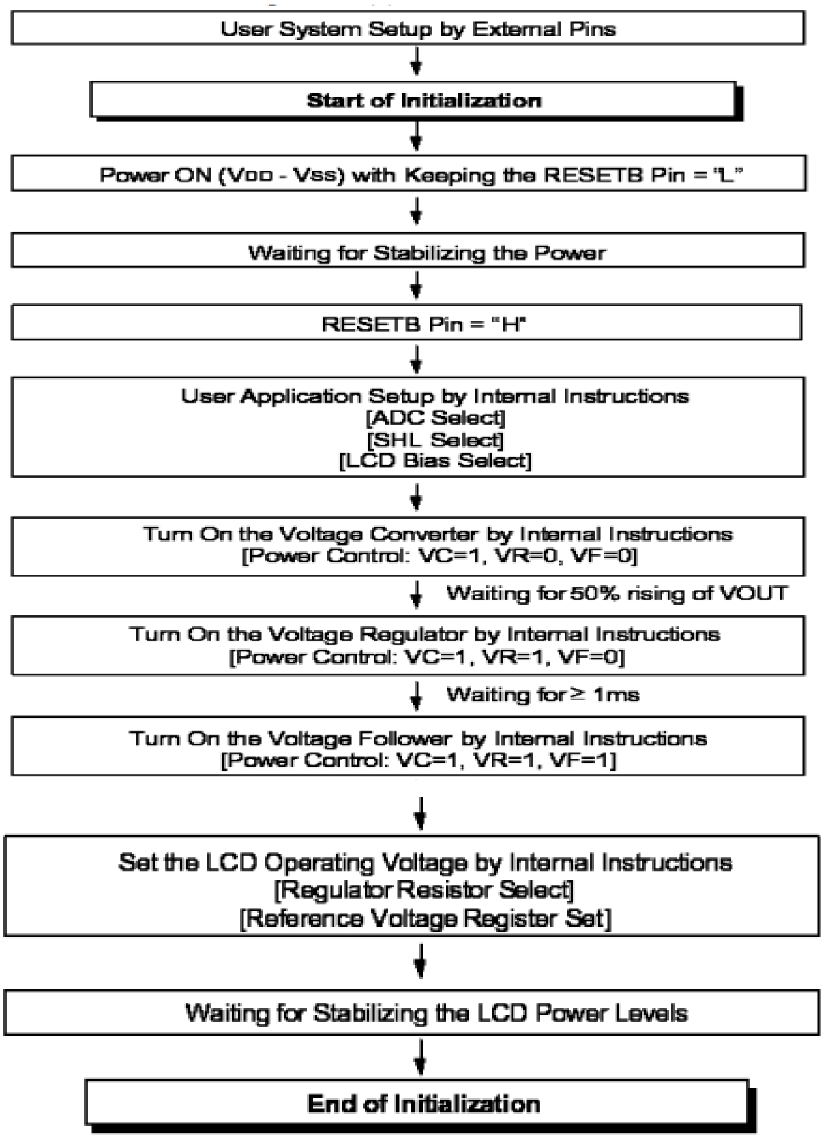
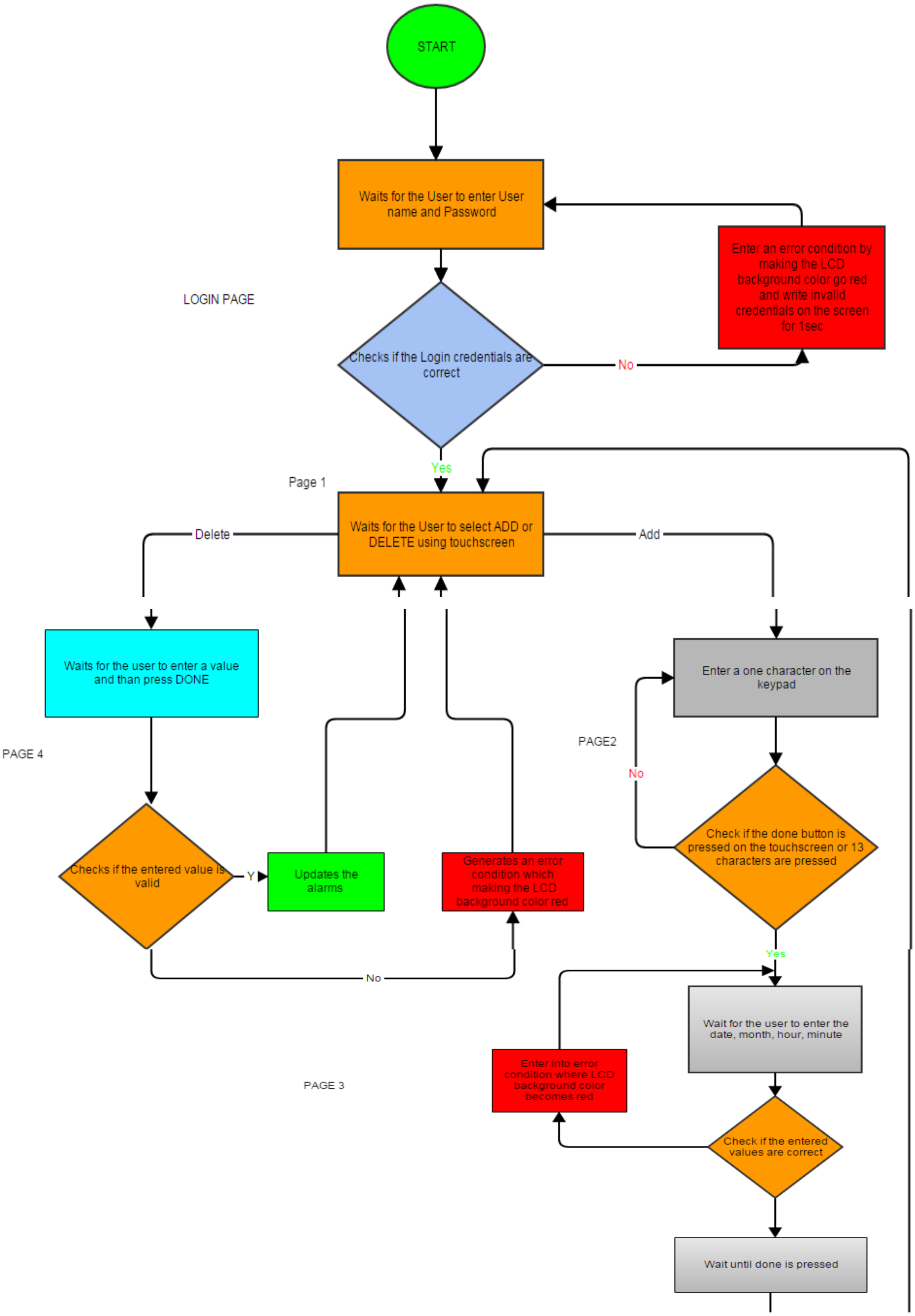

User System Setup by External Pins
Start of Initialization
Power ON (VDD - VSS) with Keeping the RESETB Pin = "L"
Waiting for Stabilizing the Power
RESETB Pin = "H"
User Application Setup by Internal Instructions
[ADC Select]
[SHL Select]
[LCD Bias Select]
Turn On the Voltage Converter by Internal Instructions
[Power Control: VC=1, VR=0, VF=0]
Waiting for 50% rising of VOUT
Turn On the Voltage Regulator by Internal Instructions
[Power Control: VC=1, VR=1, VF=0]
Waiting for ≥ 1ms
Turn On the Voltage Follower by Internal Instructions
[Power Control: VC=1, VR=1, VF=1]
Set the LCD Operating Voltage by Internal Instructions
[Regulator Resistor Select]
[Reference Voltage Register Set]
Waiting for Stabilizing the LCD Power Levels
End of Initialization

Figure 2.12 Initalization procedure

2.2.4 Main Program Flow chart


START
Waits for the User to enter User name and Password
LOGIN PAGE
Checks if the Login credentials are correct
No
Enter an error condition by making the LCD background color go red and write invalid credentials on the screen for 1sec
Yes
Page 1
Waits for the User to select ADD or DELETE using touchscreen
Delete
Add
Waits for the user to enter a value and than press DONE
PAGE 4
Checks if the entered value is valid
Y
Updates the alarms
Generates an error condition which making the LCD background color red
No
Enter a one character on the keypad
PAGE2
No
Check if the done button is pressed on the touchscreen or 13 characters are pressed
Yes
Wait for the user to enter the date, month, hour, minute
PAGE 3
Enter into error condition where LCD background color becomes red
Check if the entered values are correct
Wait until done is pressed

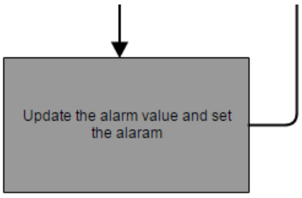


Figure 2.13 Flow chart of the main program

Page 1,Page 2 ,Page 3,Page 4 are segregated based on the change of UI on the LCD screen. Alaram count value is updated whenever the program is in page4 or page 3

CHAPTER 3

RESULT ANALYSIS

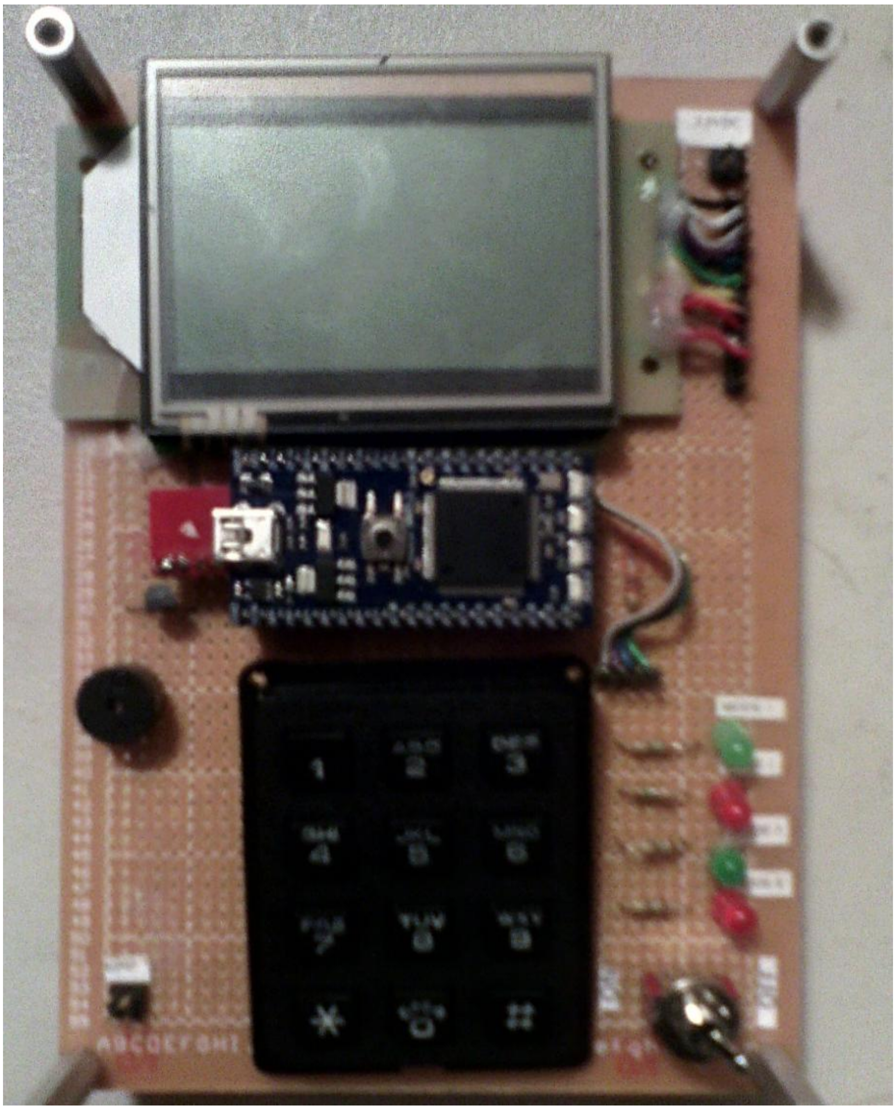


Figure 3.1 Front side of Task Scheduler

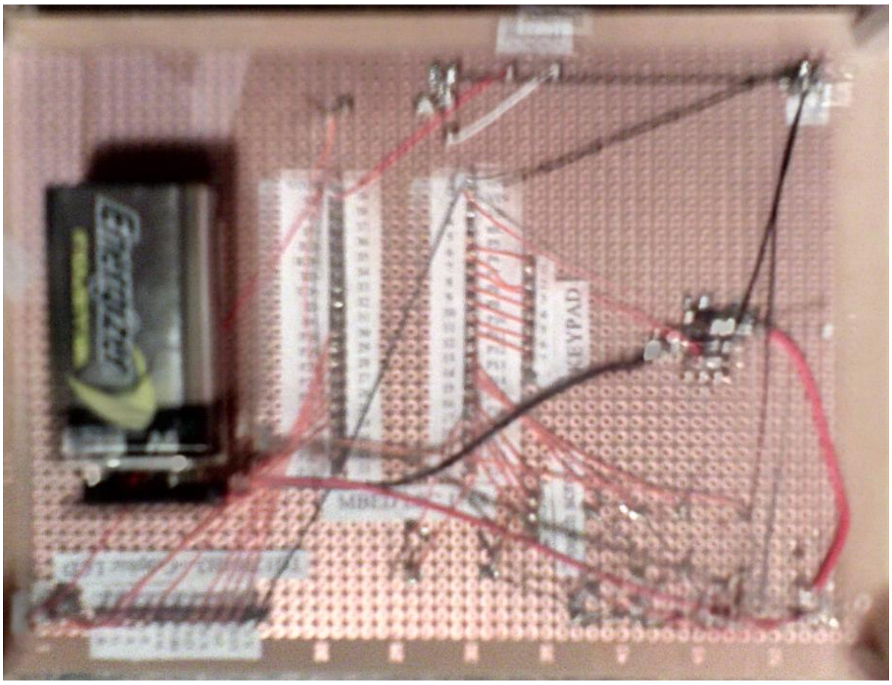


Figure 3-2 Back side of Task scheduler

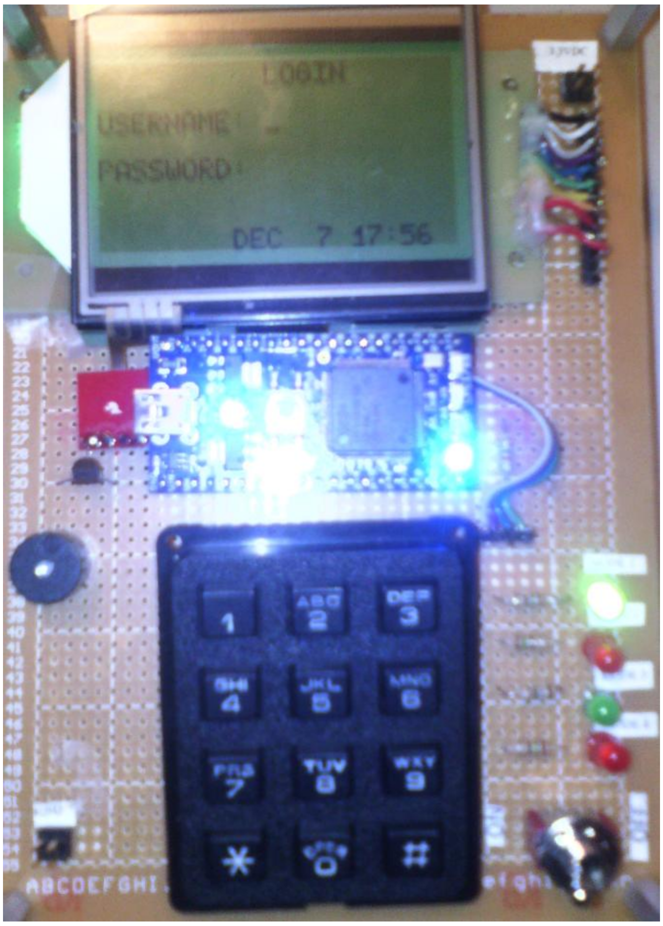


Figure 3-3 Powered on

Figure 3.3 shows the RTC on the bottom right corner on LCD and we can see that program is in login page function.

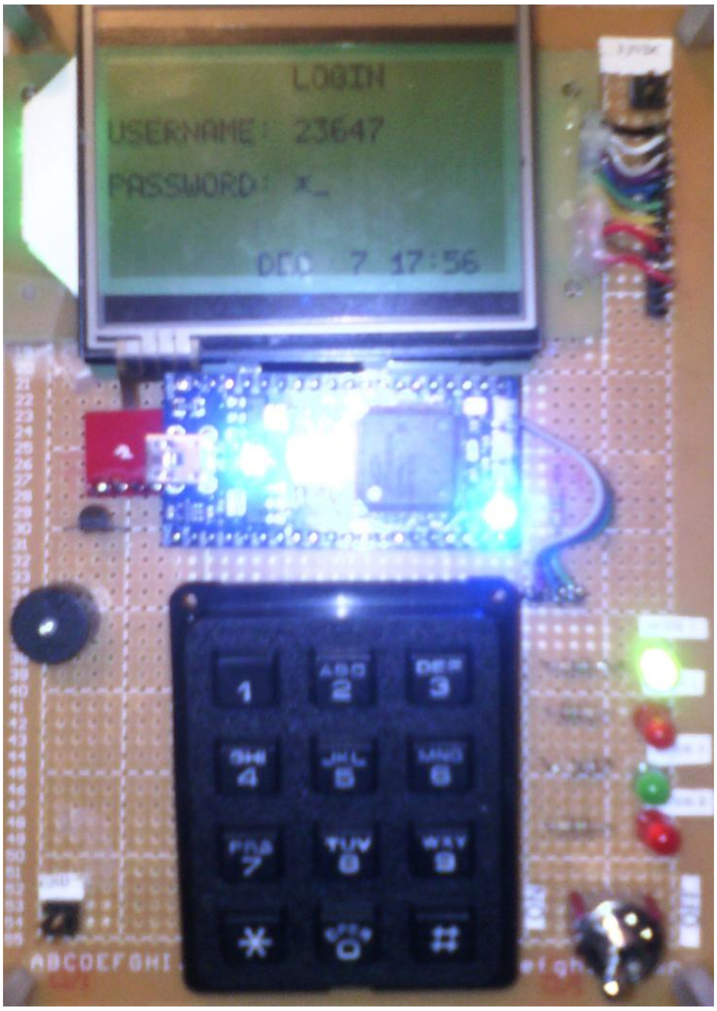


Figure 3.4 password protection

Figure 3.4 shows the feature of hiding password for security reasons

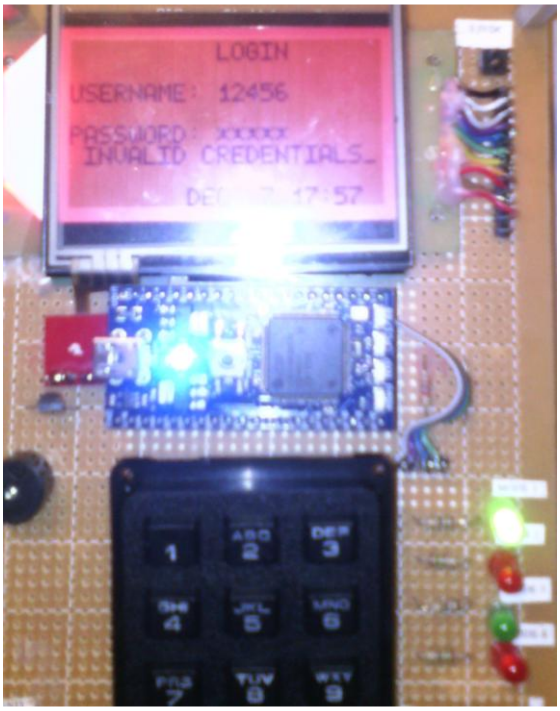


Figure 3-4 error condition

Figure 3.4 indicates the error conditions occurred by entering the invalid credentials.

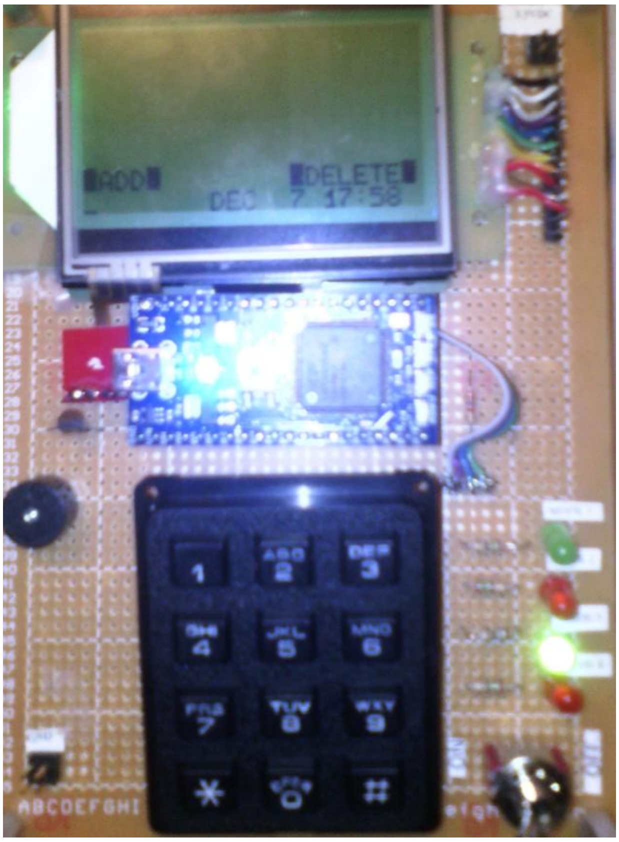


Figure 3.5 Homescreen

Figure 3.5 shows the Home screen

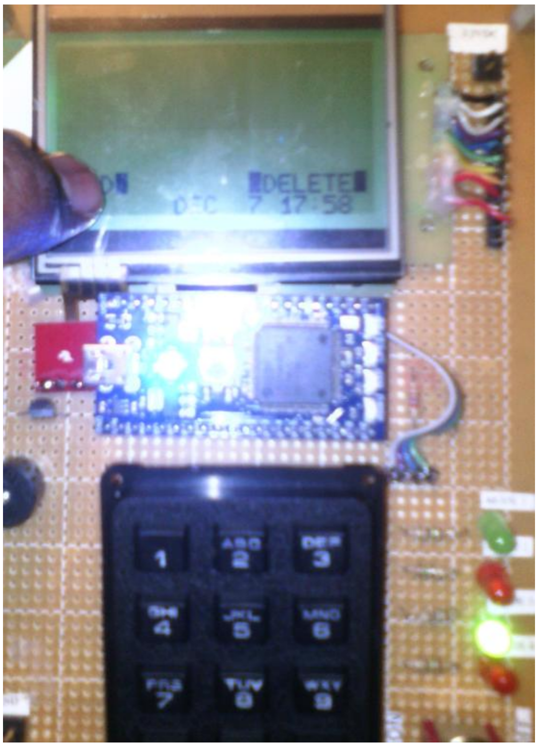


Figure 3.6 Adding task-1

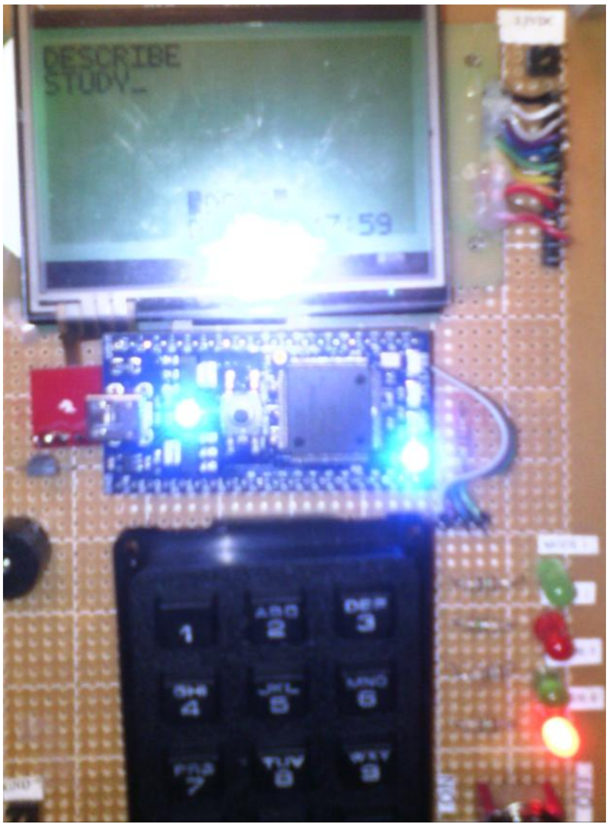


Figure 3.7 Describing a task

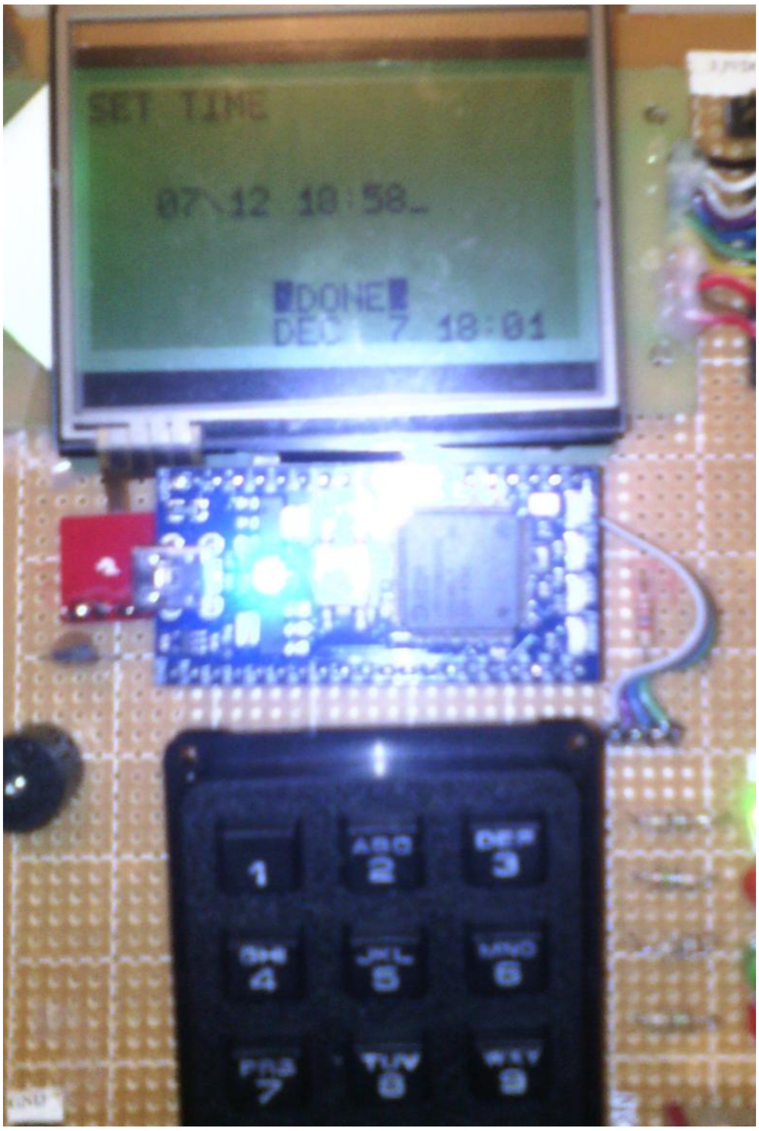


Figure 3.8 Adding time for task

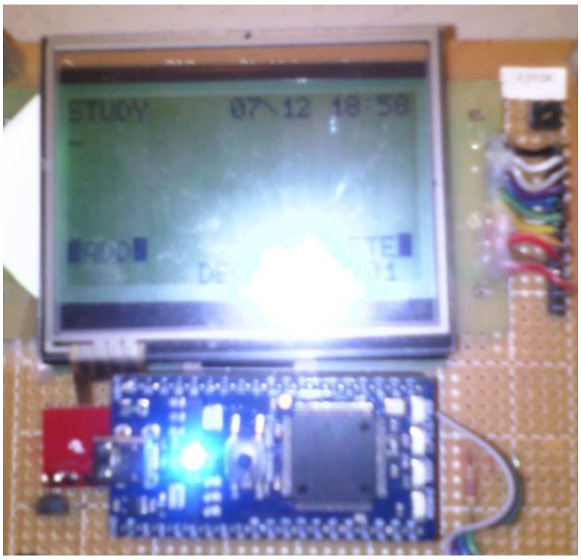


Figure 3.9 Home screen indicating the status

Figure 3.7, Figure 3.8 Figure 3.9 Figure 3.10 tells how task can be added and how it reflects on the homescreen

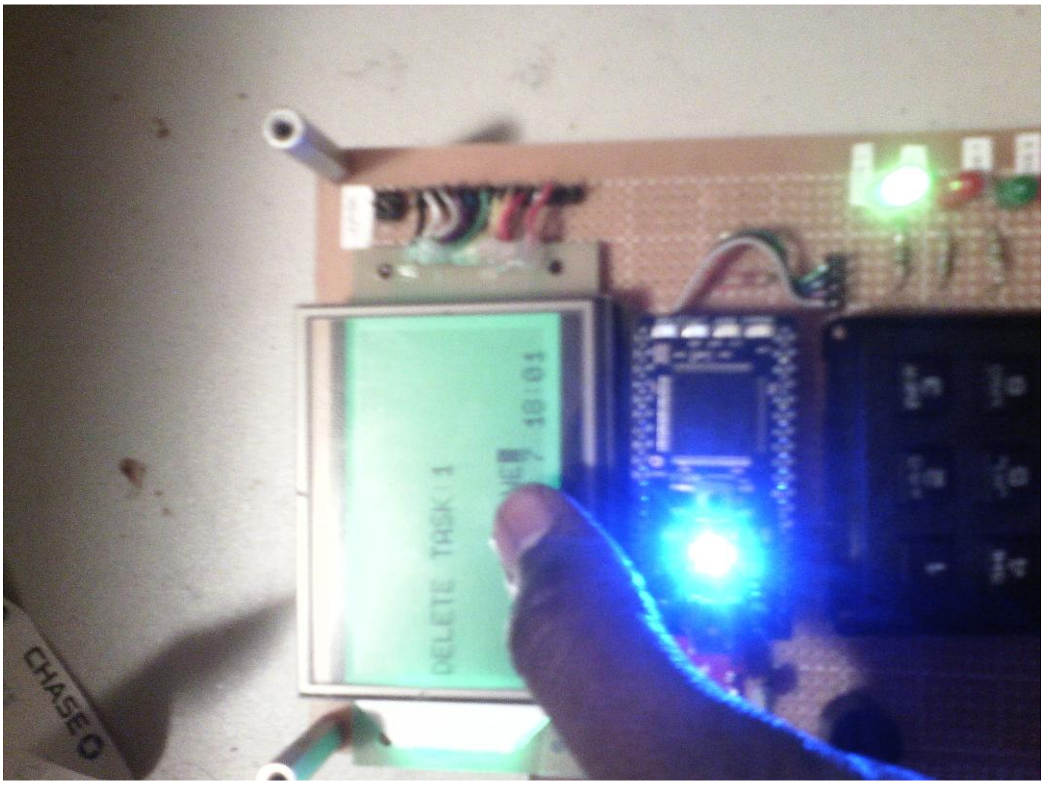


Figure 3.19 Deleting an alaram

Figure 3.10 shows how to delete a task

Chapter 4

CONCLUSIONS

As I never worked on ARM controller before, it was a good learning experience. I ran into many issues in the case of the LCD, touchscreen, which helped me to gain experience on this modules.

ARM architecture is a complex architecture, to understand the entire architecture more time and more projects on it are required. Using the inbuilt libraries would not help in learning the architecture it just speeds up the process without understanding what's going on at the register level.

Chapter 5
FUTURE DEVELOPMENTS

Currently the Task scheduler is limited to 5 alarms. I would like to increase it to at least 100 and also configure a better library for touch commands. This would help in sliding the screen to view all the options on the screen easily. I also intend to develop a PCB for the LPC1768 instead of using a development by Mbed.

Chapter 6
ACKNOWLEDGMENTS

Apart from my efforts, the success of my project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I would also like to take this opportunity to thank Professor Linden Mcclure for designing such a great course which has helped in building our basics to a higher level. I would like to thank the TA's and also my fellow classmates for helping me time and time whenever I required it.

I perceive this opportunity as a big milestone in my career development. I will strive to use the gained skills and knowledge in the best possible way.I hope to continue communication with all of you in the future.

Chapter 7

References

1. ARM controller assembly coding--
http://www.eecs.umich.edu/courses/eecs373/labsW14/refs/M3%20Guide.pdf

2. Schematics of mbed development board

https://developer.mbed.org/media/uploads/chris/mbed-005.1.pdf

3. ARM architecture

www.wikipidea.com

4. ARM cortex M3 C coding

www.embeddedguru.com

5. ARM cortex M3 Coding ---www.duolus.com

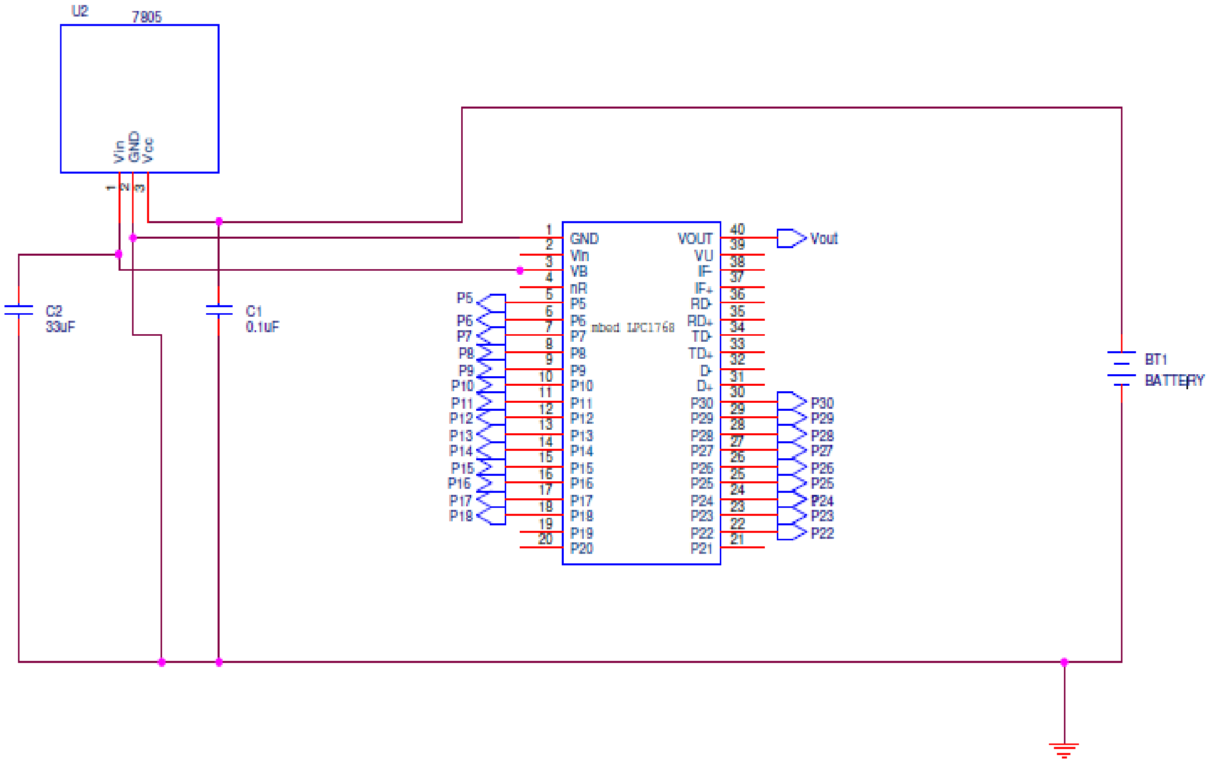
Chapter 8

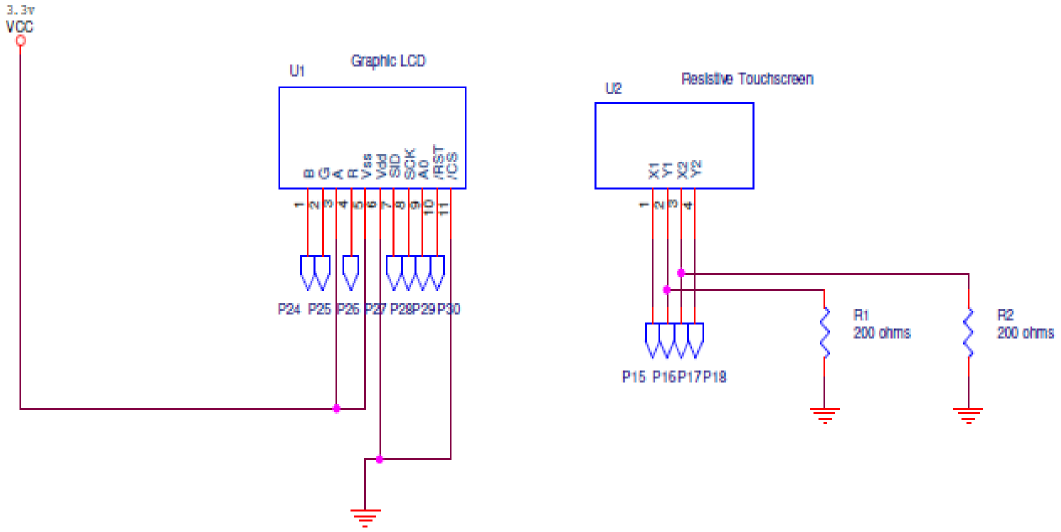
APPENDICES

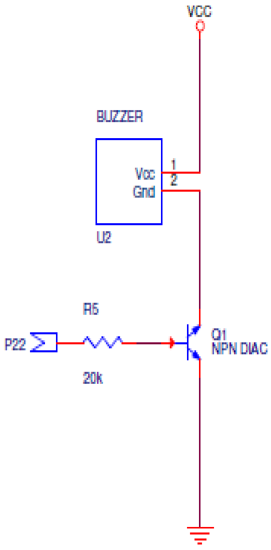
8.1 Appendix-Bill of Materials

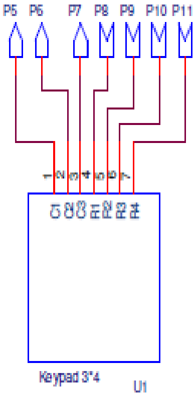
Part Description	**Source**	**Cost**
LPC1768 development board by mbed	Digikey	$60.00
Graphic LCD 128*64	Adafruit	$17.00
Resistive Touch Screen	Sparkfun	$9.00
Buzzer 5VDC	Adafruit	$0.40
Voltage regulator LM7805	EE store	$2.50
33uf ceramic Capacitor	EE store	$0.50
0.1uF ceramic capacitor	EE store	$0.50
SPST	EE store	$1.00
Stackable headers	Sparkfun	$2.50
9V DC battery	EE store	$3.50
222A npn transistor	ESD kit	$0.00
Four LED’s (2R, 2G)	ESD kit	$0.00
Male headers	ESD kit	$0.00
18k ohms ¼ watt	ESD lab	$0.00
200 ohms ¼ watt	ESD lab	$0.00
Total		**$95.00**

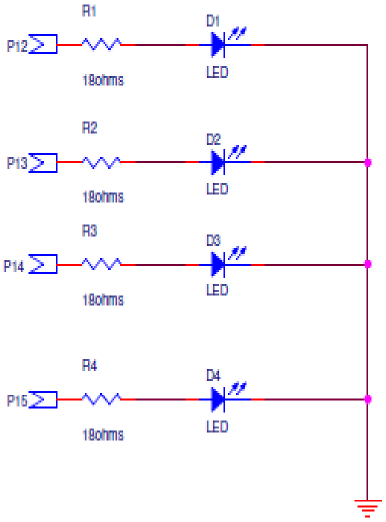
8.2 Appendix –Schematics






VCC
BUZZER
Vcc
Gnd
1
2
U2
R5
P22
20k
Q1
NPN DIAC


P5
P6
P7
P8
P9
P10
P11
1
2
3
4
5
6
7
C1
C2
C3
R1
R2
R3
R4
Keypad 3*4
U1


R1
P12
18ohms
D1
LED
R2
P13
18ohms
D2
LED
R3
P14
18ohms
D3
LED
R4
P15
18ohms
D4
LED

8.3 Appendix – Firmware source code

```
/*----------------------------------------------------------------------------------------------------------------
------------------------------------------------------
 * Raghunath Reddy
 * ECEN 5613, Task scheduler
 * Fall 2016,Prof. Mc Clure
 * University of Colorado at Boulder
 * ---------------------------------------
 * This file is an application specific file which uses other .h files to control LCD,
keypad,touch screen.This is used to make dedicated embedded device name
 task scheduler to work
 embed.h is developed by Mihail Stoyano ,which i am using for system initalization and RTC
 * --------------------------------------------------------------------------------------------------------------
---------------------------------------------------- */
#include "mbed.h"                                        //Already present in the mbed
complier used for RTC,ADC
#include "touchscreen.h"
#include "lcdes.h"
#include "keypadesd.h"
Ticker lcd_time1;                                        //repetetive interrupt


/* Prototypes -------------------------------------------------- */
 /*--------------------------------------------------------------------------------------------------------------
-------------------------------------------------------
  * void login
  * Purpose:     Waits for the user to input the data and the matches with already existing
username and password if it matches than goes into page or else erroer
    condition is displayed
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------ */
void login();

 /*--------------------------------------------------------------------------------------------------------------
-------------------------------------------------------
  * void page1
  * Purpose:     Page 1 is the home screen where the description of all the alarams , time for
the alaram is also displayed
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------ */
void page1();


 /*--------------------------------------------------------------------------------------------------------------
-------------------------------------------------------
  * void login_user
  * Purpose:     Checks if  the enetered data matches whith the existing dat of username and
password
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void login_user();

 /*---------------------------------------------------------------------------------------------------------
------------------------------------------------------
  * void initalize_rtc
  * Purpose:     This is written once to set the RTC
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void initialize_rtc();


 /*---------------------------------------------------------------------------------------------------------
------------------------------------------------------
  * void hide_character
  * Purpose:     While entering the password after a certain delay the character is turned to *
for security reasons
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void hide_character();

 /*---------------------------------------------------------------------------------------------------------
------------------------------------------------------
  * void page2
  * Purpose:     This page is waits for the user to enter the descripeion of the task
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void page2();

 /*---------------------------------------------------------------------------------------------------------
------------------------------------------------------
  * void page3
  * Purpose:     This page allows tghe user to enter the time and day for the alaram
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */

void page3();

/*----------------------------------------------------------------------------------------------------------
-----------------------------------------------------
  * void page3
  * Purpose:     This page allows tghe user to enter the time and day for the alaram
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------------------------ */
void page4();
void test_buzzer();
void adc_check_done();

/*---------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------
  * void page3
  * Purpose:     Below all the functions named write are functions to write into LCD example:
write_error writes the word error on the LCD
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------------------------ */
void login_write_error();
void login_write();
void login_write_login();
void login_write_username();
void login_write_password();
void lcd_write_string();
void lcd_write_add();
void lcd_write_delete();
void lcd_write_done();
void lcd_write_describe();
void lcd_write_description();
void lcd_write_description2();
void lcd_write_description3();
void lcd_write_description4();
void lcd_write_description5();
void lcd_write_settime();
void lcd_write_timeformat();
void lcd_write_reset();
void lcd_write_time1();
void lcd_write_time2();
void lcd_write_time3();
void lcd_write_time4();
void lcd_write_time5();
void lcd_write_deletetask();


int main()
{
  initialize_lcd_pinouts();
  initialize_lcd();
  keypad_init();
  lcd_time_display();
  lcd_time1.attach(&lcd_time_display,60.0);
  login();
    while(1)
    {
    page1();
    if(touchpressed_add_flag==1)
    {
    page2();
    page3();
    }
    else if(touchpressed_delete_flag==1)
    {
    page4();
    }
    }
  }
}

void page4()
{
      unsigned char i;

      changemode=0;                                    // change mode to zero
      LPC_GPIO0 -> FIOSET |= (1<<17);                  //Mode 0 led set
      LPC_GPIO0 -> FIOCLR |= (1<<15);                  //Mode 1 Led reset
      LPC_GPIO0 -> FIOCLR |= (1<<16);                  //Mode 2 Led reset
      LPC_GPIO2 -> FIOCLR |= (1<<3);                   //Mode 3 Led reset
      //alaram1_lcd_flag = 0;
      lcd_clear1();
      pagevalue=6;
      column=48;
      lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      lcd_write_done();
      pagevalue=3;
      column=00;
      lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      lcd_write_deletetask();                          //deletetask

     for(keypress_count=0;keypress_count<1;keypress_count++)
      {
       if (keypress_count==0)
       {
         lcd_cursorblink_flag=1;
       }
        key_press();                          //wait  till the the number of the task to be deleted
entered
        if (keypress_count==0)
       {
         lcd_cursorblink_flag=0;
       }
       alaram_time_delete = glo_line;
      }

       alaram_time_delete = (alaram_time_delete-0x30);          //converting into decimal
     //  printf("alaram time1_delete is %d",alaram_time_delete);
      if (alaram_time_delete > alaram_count)                 //corner conditions id the user wrond
number
      {
         lcd_color_red();
         wait(1);
         lcd_color_green();
         goto endpage4;
      }
       // From here this functions is similat to the function in the LCds.h named
ldc_time_display()
       if(alaram_time_delete==1)
       {

            if(alaram_count == 1)
            {
             alaram1_lcd_flag = 0;
             alaram1sec=0;
             alaram_count--;
            }
           if(alaram_count == 2)
           {
            desc_count=desc_count2;
            for(i=0;i<desc_count2;i++)
            {
            alaram_des_1[i]=alaram_des_2[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_1_1[i]=alaram_time_2_1[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_1_2[i]=alaram_time_2_2[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_1_3[i]=alaram_time_2_3[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_1_4[i]=alaram_time_2_4[i];
            }
            alaram_time1_h=alaram_time2_h;
            alaram_time1_m=alaram_time2_m;
```

```
 alaram_time1_d=alaram_time2_d;
 alaram_time1_o=alaram_time2_o;
 alaram_count--;
 alaram2sec=0;
 alaram2_lcd_flag = 0;
 alaram1();
}
 if(alaram_count == 3)
{
 desc_count=desc_count2;
 for(i=0;i<desc_count2;i++)
 {
 alaram_des_1[i]=alaram_des_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_1[i]=alaram_time_2_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_2[i]=alaram_time_2_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_3[i]=alaram_time_2_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_4[i]=alaram_time_2_4[i];
 }
 alaram_time1_h=alaram_time2_h;
 alaram_time1_m=alaram_time2_m;
 alaram_time1_d=alaram_time2_d;
 alaram_time1_o=alaram_time2_o;


 desc_count2=desc_count3;
 for(i=0;i<desc_count3;i++)
 {
 alaram_des_2[i]=alaram_des_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_1[i]=alaram_time_3_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_2[i]=alaram_time_3_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_3[i]=alaram_time_3_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_4[i]=alaram_time_3_4[i];
 }
 alaram_time2_h=alaram_time3_h;
 alaram_time2_m=alaram_time3_m;
 alaram_time2_d=alaram_time3_d;
 alaram_time2_o=alaram_time3_o;
 alaram_count--;
 alaram3sec=0;
 alaram1();
 alaram2();
 alaram3_lcd_flag = 0;
}


 if(alaram_count == 4)
{
 desc_count=desc_count2;
 for(i=0;i<desc_count2;i++)
 {
 alaram_des_1[i]=alaram_des_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_1[i]=alaram_time_2_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_2[i]=alaram_time_2_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_3[i]=alaram_time_2_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_4[i]=alaram_time_2_4[i];
 }
 alaram_time1_h=alaram_time2_h;
 alaram_time1_m=alaram_time2_m;
 alaram_time1_d=alaram_time2_d;
 alaram_time1_o=alaram_time2_o;


 desc_count2=desc_count3;
 for(i=0;i<desc_count3;i++)
```

```
{
alaram_des_2[i]=alaram_des_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_1[i]=alaram_time_3_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_2[i]=alaram_time_3_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_3[i]=alaram_time_3_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_4[i]=alaram_time_3_4[i];
}
alaram_time2_h=alaram_time3_h;
alaram_time2_m=alaram_time3_m;
alaram_time2_d=alaram_time3_d;
alaram_time2_o=alaram_time3_o;


desc_count3=desc_count4;
for(i=0;i<desc_count4;i++)
{
alaram_des_3[i]=alaram_des_4[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_1[i]=alaram_time_4_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_2[i]=alaram_time_4_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_3[i]=alaram_time_4_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_4[i]=alaram_time_4_4[i];
}
alaram_time3_h=alaram_time4_h;
alaram_time3_m=alaram_time4_m;
alaram_time3_d=alaram_time4_d;
alaram_time3_o=alaram_time4_o;
```

```
 alaram_count--;
 alaram4sec=0;
 alaram1();
 alaram2();
 alaram3();
 alaram4_lcd_flag = 0;
}

 if(alaram_count == 5)
{
 desc_count=desc_count2;
 for(i=0;i<desc_count2;i++)
 {
 alaram_des_1[i]=alaram_des_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_1[i]=alaram_time_2_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_2[i]=alaram_time_2_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_3[i]=alaram_time_2_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_4[i]=alaram_time_2_4[i];
 }
 alaram_time1_h=alaram_time2_h;
 alaram_time1_m=alaram_time2_m;
 alaram_time1_d=alaram_time2_d;
 alaram_time1_o=alaram_time2_o;


 desc_count2=desc_count3;
 for(i=0;i<desc_count3;i++)
 {
 alaram_des_2[i]=alaram_des_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_1[i]=alaram_time_3_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_2[i]=alaram_time_3_2[i];
 }
```

```
for(i=0;i<2;i++)
{
 alaram_time_2_3[i]=alaram_time_3_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_4[i]=alaram_time_3_4[i];
}
alaram_time2_h=alaram_time3_h;
alaram_time2_m=alaram_time3_m;
alaram_time2_d=alaram_time3_d;
alaram_time2_o=alaram_time3_o;


desc_count3=desc_count4;
for(i=0;i<desc_count4;i++)
{
alaram_des_3[i]=alaram_des_4[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_1[i]=alaram_time_4_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_2[i]=alaram_time_4_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_3[i]=alaram_time_4_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_4[i]=alaram_time_4_4[i];
}
alaram_time3_h=alaram_time4_h;
alaram_time3_m=alaram_time4_m;
alaram_time3_d=alaram_time4_d;
alaram_time3_o=alaram_time4_o;


desc_count4=desc_count5;
for(i=0;i<desc_count5;i++)
{
alaram_des_4[i]=alaram_des_5[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_1[i]=alaram_time_5_1[i];
}
```

```
     for(i=0;i<2;i++)
     {
      alaram_time_4_2[i]=alaram_time_5_2[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_3[i]=alaram_time_5_3[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_4[i]=alaram_time_5_4[i];
     }
     alaram_time4_h=alaram_time5_h;
     alaram_time4_m=alaram_time5_m;
     alaram_time4_d=alaram_time5_d;
     alaram_time4_o=alaram_time5_o;
     alaram_count--;
     alaram5sec=0;
     alaram1();
     alaram2();
     alaram3();
     alaram4();
     alaram5_lcd_flag = 0;
    }



}




       if(alaram_time_delete==2)
{

    if(alaram_count == 2)
    {
     alaram_count--;
     alaram2sec=0;
     alaram2_lcd_flag = 0;
    // alaram1();
    }
     if(alaram_count == 3)
    {

     desc_count2=desc_count3;
     for(i=0;i<desc_count3;i++)
     {
     alaram_des_2[i]=alaram_des_3[i];
     }
```

```
 for(i=0;i<2;i++)
 {
  alaram_time_2_1[i]=alaram_time_3_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_2[i]=alaram_time_3_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_3[i]=alaram_time_3_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_4[i]=alaram_time_3_4[i];
 }
 alaram_time2_h=alaram_time3_h;
 alaram_time2_m=alaram_time3_m;
 alaram_time2_d=alaram_time3_d;
 alaram_time2_o=alaram_time3_o;
 alaram_count--;
 alaram3sec=0;
 //alaram1();
 alaram2();
 alaram3_lcd_flag = 0;
}


 if(alaram_count == 4)
{
 desc_count2=desc_count3;
 for(i=0;i<desc_count3;i++)
 {
 alaram_des_2[i]=alaram_des_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_1[i]=alaram_time_3_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_2[i]=alaram_time_3_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_3[i]=alaram_time_3_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_4[i]=alaram_time_3_4[i];
 }
 alaram_time2_h=alaram_time3_h;
 alaram_time2_m=alaram_time3_m;
 alaram_time2_d=alaram_time3_d;
 alaram_time2_o=alaram_time3_o;


 desc_count3=desc_count4;
 for(i=0;i<desc_count4;i++)
 {
 alaram_des_3[i]=alaram_des_4[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_1[i]=alaram_time_4_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_2[i]=alaram_time_4_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_3[i]=alaram_time_4_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_4[i]=alaram_time_4_4[i];
 }
 alaram_time3_h=alaram_time4_h;
 alaram_time3_m=alaram_time4_m;
 alaram_time3_d=alaram_time4_d;
 alaram_time3_o=alaram_time4_o;
 alaram_count--;
 alaram4sec=0;
// alaram1();
 alaram2();
 alaram3();
 alaram4_lcd_flag = 0;
}

 if(alaram_count == 5)
{

 desc_count2=desc_count3;
 for(i=0;i<desc_count3;i++)
 {
 alaram_des_2[i]=alaram_des_3[i];
 }
 for(i=0;i<2;i++)
 {
```

```
 alaram_time_2_1[i]=alaram_time_3_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_2[i]=alaram_time_3_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_3[i]=alaram_time_3_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_4[i]=alaram_time_3_4[i];
}
alaram_time2_h=alaram_time3_h;
alaram_time2_m=alaram_time3_m;
alaram_time2_d=alaram_time3_d;
alaram_time2_o=alaram_time3_o;


desc_count3=desc_count4;
for(i=0;i<desc_count4;i++)
{
alaram_des_3[i]=alaram_des_4[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_1[i]=alaram_time_4_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_2[i]=alaram_time_4_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_3[i]=alaram_time_4_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_4[i]=alaram_time_4_4[i];
}
alaram_time3_h=alaram_time4_h;
alaram_time3_m=alaram_time4_m;
alaram_time3_d=alaram_time4_d;
alaram_time3_o=alaram_time4_o;


desc_count4=desc_count5;
for(i=0;i<desc_count5;i++)
{
    alaram_des_4[i]=alaram_des_5[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_4_1[i]=alaram_time_5_1[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_4_2[i]=alaram_time_5_2[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_4_3[i]=alaram_time_5_3[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_4_4[i]=alaram_time_5_4[i];
    }
    alaram_time4_h=alaram_time5_h;
    alaram_time4_m=alaram_time5_m;
    alaram_time4_d=alaram_time5_d;
    alaram_time4_o=alaram_time5_o;



    alaram_count--;
    alaram5sec=0;
    //alaram1();
    alaram2();
    alaram3();
    alaram4();
    alaram5_lcd_flag = 0;
   }
}


 if(alaram_time_delete==3)
{

    if(alaram_count == 3)
   {
    alaram_count--;
    alaram3sec=0;
    //alaram1();
    //alaram2();
    alaram3_lcd_flag = 0;
   }
    if(alaram_count == 4)
   {
```

```
 desc_count3=desc_count4;
 for(i=0;i<desc_count4;i++)
 {
 alaram_des_3[i]=alaram_des_4[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_1[i]=alaram_time_4_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_2[i]=alaram_time_4_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_3[i]=alaram_time_4_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_4[i]=alaram_time_4_4[i];
 }
 alaram_time3_h=alaram_time4_h;
 alaram_time3_m=alaram_time4_m;
 alaram_time3_d=alaram_time4_d;
 alaram_time3_o=alaram_time4_o;
 alaram_count--;
 alaram4sec=0;
// alaram1();
// alaram2();
 alaram3();
 alaram4_lcd_flag = 0;
}

 if(alaram_count == 5)
{



 desc_count3=desc_count4;
 for(i=0;i<desc_count4;i++)
 {
 alaram_des_3[i]=alaram_des_4[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_1[i]=alaram_time_4_1[i];
 }
 for(i=0;i<2;i++)
 {
```

```
 alaram_time_3_2[i]=alaram_time_4_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_3[i]=alaram_time_4_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_4[i]=alaram_time_4_4[i];
}
alaram_time3_h=alaram_time4_h;
alaram_time3_m=alaram_time4_m;
alaram_time3_d=alaram_time4_d;
alaram_time3_o=alaram_time4_o;


desc_count4=desc_count5;
for(i=0;i<desc_count5;i++)
{
alaram_des_4[i]=alaram_des_5[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_1[i]=alaram_time_5_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_2[i]=alaram_time_5_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_3[i]=alaram_time_5_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_4[i]=alaram_time_5_4[i];
}
alaram_time4_h=alaram_time5_h;
alaram_time4_m=alaram_time5_m;
alaram_time4_d=alaram_time5_d;
alaram_time4_o=alaram_time5_o;



alaram_count--;
alaram5sec=0;
//alaram1();
//alaram2();
alaram3();
alaram4();
```

```
     alaram5_lcd_flag = 0;
    }
}




      if(alaram_time_delete==4)
{

     if(alaram_count == 4)
    {


     alaram_count--;
     alaram4sec=0;
    // alaram1();
    // alaram2();
    // alaram3();
     alaram4_lcd_flag = 0;
    }

     if(alaram_count == 5)
    {


     desc_count4=desc_count5;
     for(i=0;i<desc_count5;i++)
     {
     alaram_des_4[i]=alaram_des_5[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_1[i]=alaram_time_5_1[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_2[i]=alaram_time_5_2[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_3[i]=alaram_time_5_3[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_4[i]=alaram_time_5_4[i];
     }
     alaram_time4_h=alaram_time5_h;
     alaram_time4_m=alaram_time5_m;
     alaram_time4_d=alaram_time5_d;
```

```
            alaram_time4_o=alaram_time5_o;



            alaram_count--;
            alaram5sec=0;
            //alaram1();
            //alaram2();
            //alaram3();
            alaram4();
            alaram5_lcd_flag = 0;
           }
       }
       if(alaram_time_delete==5)
       {
            if(alaram_count == 5)
           {
            alaram_count--;
            alaram5sec=0;
            alaram5_lcd_flag = 0;
           }
       }
       xcor=0;
       ycor=0;
       touchpressed_done2_flag=0;
       while(touchpressed_done2_flag == 0)
       {
        for(i=0;i<5;i++)
        {
          adc_read();
          if(((xcor > 0x5200 && xcor< 0x7400)&&(ycor >0x7100 && ycor< 0x7C00)))
          {
           printf("\n\rsuccess\n\r");
           touchpressed_done2_flag=1;
           //goto endloop1;
          }
        }
       }
       endpage4:
       printf("\n\r i am here 6\n\r");

       }


void test_buzzer()
{
    LPC_GPIO2 -> FIOSET |= (1<<4);
    wait(2);
    LPC_GPIO2 -> FIOCLR |= (1<<4);
}
```

```
void initialize_rtc()
{
set_time(1449510960);  // seconds for dec 7 5:47 pm
}

void alaram1()
{

       struct tm t;
        t.tm_sec = 00;    // 0-59
        t.tm_min = alaram_time1_m;    // 0-59
       t.tm_hour = alaram_time1_h;   // 0-23
       t.tm_mday = alaram_time1_d;;   // 1-31
       t.tm_mon = (alaram_time1_o - 1);     // 0-11
       t.tm_year = 115;  // year since 1900
       time_t seconds = mktime(&t);
       alaram1sec= seconds;
       printf("Minutes = %d\n", alaram_time1_m);
       printf("Hours = %d\n", alaram_time1_h);
       printf("Time as seconds since January 1, 1970 33 = %d\n", seconds);
       printf("alaram1 value is %d \n\r ",alaram1);
       alaram1_flag=1;

}

void alaram2()
{

       struct tm t;
        t.tm_sec = 00;    // 0-59
        t.tm_min = alaram_time2_m;    // 0-59
       t.tm_hour = alaram_time2_h;   // 0-23
       t.tm_mday = alaram_time2_d;;   // 1-31
       t.tm_mon = (alaram_time2_o - 1);     // 0-11
       t.tm_year = 115;  // year since 1900
       time_t seconds = mktime(&t);
       alaram2sec= seconds;
       printf("Minutes = %d\n", alaram_time2_m);
       printf("Hours = %d\n", alaram_time2_h);
       printf("Time as seconds since January 1, 1970 33 = %d\n", seconds);
       printf("alaram1 value is %d \n\r ",alaram2);
       alaram2_flag=1;

}


void alaram3()
{
       struct tm t;
        t.tm_sec = 00;    // 0-59
        t.tm_min = alaram_time3_m;    // 0-59
       t.tm_hour = alaram_time3_h;   // 0-23
       t.tm_mday = alaram_time3_d;;   // 1-31
       t.tm_mon = (alaram_time3_o - 1);     // 0-11
       t.tm_year = 115;  // year since 1900
       time_t seconds = mktime(&t);
       alaram3sec= seconds;
       printf("Minutes = %d\n", alaram_time3_m);
       printf("Hours = %d\n", alaram_time3_h);
       printf("Time as seconds since January 1, 1970 33 = %d\n", seconds);
       printf("alaram1 value is %d \n\r ",alaram3);
       alaram3_flag=1;

}

void alaram4()
{

       struct tm t;
        t.tm_sec = 00;    // 0-59
        t.tm_min = alaram_time4_m;    // 0-59
       t.tm_hour = alaram_time4_h;   // 0-23
       t.tm_mday = alaram_time4_d;;   // 1-31
       t.tm_mon = (alaram_time4_o - 1);     // 0-11
       t.tm_year = 115;  // year since 1900
       time_t seconds = mktime(&t);
       alaram4sec= seconds;
       printf("Minutes = %d\n", alaram_time4_m);
       printf("Hours = %d\n", alaram_time4_h);
       printf("Time as seconds since January 1, 1970 33 = %d\n", seconds);
       printf("alaram1 value is %d \n\r ",alaram4);
       alaram4_flag=1;

}


void alaram5()
{

       struct tm t;
        t.tm_sec = 00;    // 0-59
        t.tm_min = alaram_time5_m;    // 0-59
       t.tm_hour = alaram_time5_h;   // 0-23
       t.tm_mday = alaram_time5_d;;   // 1-31
       t.tm_mon = (alaram_time5_o - 1);     // 0-11
       t.tm_year = 115;  // year since 1900
       time_t seconds = mktime(&t);
       alaram5sec= seconds;
       printf("Minutes = %d\n", alaram_time5_m);
       printf("Hours = %d\n", alaram_time5_h);
       printf("Time as seconds since January 1, 1970 33 = %d\n", seconds);
       printf("alaram1 value is %d \n\r ",alaram5);
       alaram5_flag=1;

}







void page1()
{     //
       page1start:
       lcd_clear1();
       pagevalue=6;
       column=0x00;
       lcd_cursor_column_flag=1;
       lcd_changelocation();
       lcd_cursor_column_flag=0;
       lcd_write_add();
       pagevalue=6;
       column=78;
       lcd_cursor_column_flag=1;
       lcd_changelocation();
       lcd_cursor_column_flag=0;
       lcd_write_delete();

       // from here alaramx_lcd_flag tells if the alaram has been assigned if yes than it prints out
on the LCD
       if(alaram1_lcd_flag == 1)
       {
       pagevalue=0;
       column=00;
       lcd_cursor_column_flag=1;
       lcd_changelocation();
       lcd_cursor_column_flag=0;
       lcd_write_description(); //write the description on the LCD
       pagevalue=0;
       column=60;
       lcd_cursor_column_flag=1;
       lcd_changelocation();
       lcd_cursor_column_flag=0;
       lcd_write_time1();        //write the value of the time on the LCD
       }
```

```
if(alaram2_lcd_flag == 1)
{
pagevalue=1;
column=00;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_description2();
pagevalue=1;
column=60;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_time2();
}

 if(alaram3_lcd_flag == 1)
{

pagevalue=2;
column=00;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_description3();
pagevalue=2;
column=60;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_time3();

}

if(alaram4_lcd_flag == 1)
{

pagevalue=3;
column=00;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_description4();
pagevalue=3;
column=60;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_time4();
```

```
    }


    if(alaram5_lcd_flag == 1)
    {

    pagevalue=4;
    column=00;
    lcd_cursor_column_flag=1;
    lcd_changelocation();
    lcd_cursor_column_flag=0;
    lcd_write_description5();
    pagevalue=4;
    column=60;
    lcd_cursor_column_flag=1;
    lcd_changelocation();
    lcd_cursor_column_flag=0;
    lcd_write_time5();

    }



    touchpressed_add_flag=0;
    touchpressed_delete_flag=0;

    while (touchpressed_add_flag==0 && touchpressed_delete_flag==0 ) //wait till either
add or delete button is touched on the screen
    {
      adc_read();
     if(((xcor > 0x1700 && xcor< 0x2F00)&&(ycor >0x7A00 && ycor< 0x8500)))
     {
      printf("\n\rsuccess\n\r");
      touchpressed_add_flag=1;
      alaram_count++;
      if(alaram_count > 5)
      {
         alaram_count=0;
      }
     }
      adc_read();
     if(((xcor > 0x7800 && xcor< 0xA600)&&(ycor >0x6A00 && ycor< 0x7D00)))
     {
      printf("\n\rsuccess\n\r");
      touchpressed_delete_flag=1;
     }
     if(lcd_refresh == 1)
     {
      lcd_refresh=0;
      goto page1start;
```

```
        }
       }
}



void page2()
{
      unsigned char i;
      touchpressed_done_flag=0;
      lcd_clear1();
      pagevalue=0;
      column=0x00;
      lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      lcd_write_describe();
      pagevalue=6;
      column=48;
      lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      lcd_write_done();
      pagevalue=1;
      column=00;
      lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      xcor=0;
      ycor=0;
      touchpressed_done_flag=0;
      wait(1);
      page2_flag=1;
      if(alaram_count ==1 )
      {
      for(keypress_count=0;keypress_count<9;keypress_count++)
      {
       key_press();
       alaram_des_1[keypress_count]= glo_line;
       if(touchpressed_done_flag==1)
       {
        goto nextpage;
       }
      }
      nextpage:
      desc_count = keypress_count;
      touchpressed_done_flag=0;
      for(i=0;i<desc_count;i++)
      {
       printf("Entered string is %c",alaram_des_1[i]);
```

```
}
}
if(alaram_count == 2 )
{
for(keypress_count=0;keypress_count<13;keypress_count++)
{
 key_press();
 alaram_des_2[keypress_count]= glo_line;
 if(touchpressed_done_flag==1)
 {
  goto nextpage2;
 }
}
nextpage2:
desc_count2 = keypress_count;
touchpressed_done_flag=0;
for(i=0;i<desc_count2;i++)
{
 printf("Entered string is %c",alaram_des_2[i]);
}
}


if(alaram_count == 3 )
{
for(keypress_count=0;keypress_count<13;keypress_count++)
{
 key_press();
 alaram_des_3[keypress_count]= glo_line;
 if(touchpressed_done_flag==1)
 {
  goto nextpage3;
  }
}
nextpage3:
desc_count3 = keypress_count;
touchpressed_done_flag=0;
for(i=0;i<desc_count3;i++)
{
 printf("Entered string is %c",alaram_des_3[i]);
}
}


if(alaram_count == 4 )
{
for(keypress_count=0;keypress_count<13;keypress_count++)
{
 key_press();
 alaram_des_4[keypress_count]= glo_line;
```

```
          if(touchpressed_done_flag==1)
          {
           goto nextpage4;
           }
         }
         nextpage4:
         desc_count4 = keypress_count;
         touchpressed_done_flag=0;
         for(i=0;i<desc_count4;i++)
         {
          printf("Entered string is %c",alaram_des_4[i]);
         }
         }

        if(alaram_count == 5 )
         {
         for(keypress_count=0;keypress_count<13;keypress_count++)
         {
          key_press();
          alaram_des_5[keypress_count]= glo_line;
          if(touchpressed_done_flag==1)
          {
           goto nextpage5;
           }
         }
         nextpage5:
         desc_count5 = keypress_count;
         touchpressed_done_flag=0;
         for(i=0;i<desc_count5;i++)
         {
          printf("Entered string is %c",alaram_des_5[i]);
         }
         }

}

void page3()
{
       unsigned char touchpressed_done_flag2;
       unsigned char i;
       page3start:
       changemode=0;
       LPC_GPIO0 -> FIOSET |= (1<<17);
       LPC_GPIO0 -> FIOCLR |= (1<<15);
       LPC_GPIO0 -> FIOCLR |= (1<<16);
       LPC_GPIO2 -> FIOCLR |= (1<<3);
       lcd_clear1();
       pagevalue=0;
       column=0x00;
       lcd_cursor_column_flag=1;
```

```
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_settime();
pagevalue=3;
column=18;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_timeformat();
pagevalue=6;
column=48;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_write_done();
pagevalue=3;
column=18;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
keypress_count=0;

if(alaram_count == 1)
{
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_1_3[keypress_count]= glo_line;
}
 alaram_time1_d = ((alaram_time_1_3[0]-0x30)*10) + (alaram_time_1_3[1]-0x30);
 printf("alaram time1_d is %d",alaram_time1_d);

pagevalue=3;
column=36;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
keypress_count=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_1_4[keypress_count]= glo_line;
}
 alaram_time1_o = ((alaram_time_1_4[0]-0x30)*10) + (alaram_time_1_4[1]-0x30);
 printf("alaram time1_o is %d",alaram_time1_o);
pagevalue=3;
column=54;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
keypress_count=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_1_1[keypress_count]= glo_line;
}
 alaram_time1_h = ((alaram_time_1_1[0]-0x30)*10) + (alaram_time_1_1[1]-0x30);
 printf("alaram time1_h is %d",alaram_time1_h);
pagevalue=3;
column=72;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;

for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
 key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_1_2[keypress_count]= glo_line;
}
 alaram_time1_m = ((alaram_time_1_2[0]-0x30)*10) + (alaram_time_1_2[1]-0x30);
 printf("alaram time1_m is %d",alaram_time1_m);
 if( (alaram_time1_m > 60) || (alaram_time1_h > 23) || (alaram_time1_d > 31) ||
(alaram_time1_o > 12))
 {
  lcd_color_red();
  wait(1);
  lcd_color_green();
  goto    page3start;
 }

 }


else if(alaram_count == 2)
{

for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_2_3[keypress_count]= glo_line;
}
 alaram_time2_d = ((alaram_time_2_3[0]-0x30)*10) + (alaram_time_2_3[1]-0x30);
 printf("alaram time2_d is %d",alaram_time2_d);
pagevalue=3;
column=36;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
keypress_count=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
  lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
  lcd_cursorblink_flag=0;
 }
 alaram_time_2_4[keypress_count]= glo_line;
}
 alaram_time2_o = ((alaram_time_2_4[0]-0x30)*10) + (alaram_time_2_4[1]-0x30);
 printf("alaram time2_o is %d",alaram_time2_o);
pagevalue=3;
column=54;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
keypress_count=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
  lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
  lcd_cursorblink_flag=0;
 }
 alaram_time_2_1[keypress_count]= glo_line;
}
 alaram_time2_h = ((alaram_time_2_1[0]-0x30)*10) + (alaram_time_2_1[1]-0x30);
 printf("alaram time2_h is %d",alaram_time2_h);
pagevalue=3;
column=72;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;

for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
  lcd_cursorblink_flag=1;
 }
 key_press();
```

```
         if (keypress_count==1)
        {
          lcd_cursorblink_flag=0;
        }
        alaram_time_2_2[keypress_count]= glo_line;
       }
        alaram_time2_m = ((alaram_time_2_2[0]-0x30)*10) + (alaram_time_2_2[1]-0x30);
        printf("alaram time2_m is %d",alaram_time2_m);

         if( (alaram_time2_m > 60) || (alaram_time2_h > 23) || (alaram_time2_d > 31) ||
(alaram_time2_o > 12))
        {
         lcd_color_red();
         wait(1);
         lcd_color_green();
         goto    page3start;
        }

        }


       else if(alaram_count == 3)
       {
       for(keypress_count=0;keypress_count<2;keypress_count++)
       {
        if (keypress_count==1)
        {
          lcd_cursorblink_flag=1;
        }
         key_press();
         if (keypress_count==1)
        {
          lcd_cursorblink_flag=0;
        }
        alaram_time_3_3[keypress_count]= glo_line;
       }
        alaram_time3_d = ((alaram_time_3_3[0]-0x30)*10) + (alaram_time_3_3[1]-0x30);
        printf("alaram time3_d is %d",alaram_time3_d);

       pagevalue=3;
       column=36;
       lcd_cursorblink_flag=1;
       lcd_cursor_column_flag=1;
       lcd_changelocation();
       lcd_cursor_column_flag=0;
       lcd_cursorblink_flag=0;
       keypress_count=0;
       for(keypress_count=0;keypress_count<2;keypress_count++)
       {
        if (keypress_count==1)
```

```
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_3_4[keypress_count]= glo_line;
}
 alaram_time3_o = ((alaram_time_3_4[0]-0x30)*10) + (alaram_time_3_4[1]-0x30);
 printf("alaram time3_o is %d",alaram_time3_o);
pagevalue=3;
column=54;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
keypress_count=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_3_1[keypress_count]= glo_line;
}
 alaram_time3_h = ((alaram_time_3_1[0]-0x30)*10) + (alaram_time_3_1[1]-0x30);
 printf("alaram time3_h is %d",alaram_time3_h);
pagevalue=3;
column=72;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;

for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
 key_press();
```

```
        if (keypress_count==1)
       {
         lcd_cursorblink_flag=0;
       }
       alaram_time_3_2[keypress_count]= glo_line;
      }
       alaram_time3_m = ((alaram_time_3_2[0]-0x30)*10) + (alaram_time_3_2[1]-0x30);
       printf("alaram time3_m is %d",alaram_time3_m);

        if( (alaram_time3_m > 60) || (alaram_time3_h > 23) || (alaram_time3_d > 31) ||
(alaram_time3_o > 12))
       {
        lcd_color_red();
        wait(1);
        lcd_color_green();
        goto    page3start;
       }
       }

    else if(alaram_count == 4)
      {
      for(keypress_count=0;keypress_count<2;keypress_count++)
      {
       if (keypress_count==1)
       {
         lcd_cursorblink_flag=1;
       }
        key_press();
        if (keypress_count==1)
       {
         lcd_cursorblink_flag=0;
       }
       alaram_time_4_3[keypress_count]= glo_line;
      }
       alaram_time4_d = ((alaram_time_4_3[0]-0x30)*10) + (alaram_time_4_3[1]-0x30);
       printf("alaram time4_d is %d",alaram_time4_d);

      pagevalue=3;
      column=36;
      lcd_cursorblink_flag=1;
      lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      lcd_cursorblink_flag=0;
      keypress_count=0;
      for(keypress_count=0;keypress_count<2;keypress_count++)
      {
       if (keypress_count==1)
       {
         lcd_cursorblink_flag=1;
```

```
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_4_4[keypress_count]= glo_line;
}
 alaram_time4_o = ((alaram_time_4_4[0]-0x30)*10) + (alaram_time_4_4[1]-0x30);
 printf("alaram time4_o is %d",alaram_time4_o);
pagevalue=3;
column=54;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
keypress_count=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_4_1[keypress_count]= glo_line;
}
 alaram_time4_h = ((alaram_time_4_1[0]-0x30)*10) + (alaram_time_4_1[1]-0x30);
 printf("alaram time4_h is %d",alaram_time4_h);
pagevalue=3;
column=72;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;

for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
 key_press();
  if (keypress_count==1)
 {
         lcd_cursorblink_flag=0;
       }
       alaram_time_4_2[keypress_count]= glo_line;
      }
       alaram_time4_m = ((alaram_time_4_2[0]-0x30)*10) + (alaram_time_4_2[1]-0x30);
       printf("alaram time4_m is %d",alaram_time4_m);
        if( (alaram_time4_m > 60) || (alaram_time4_h > 23) || (alaram_time4_d > 31) ||
(alaram_time4_o > 12))
       {
        lcd_color_red();
        wait(1);
        lcd_color_green();
        goto    page3start;
       }
       }


      else if(alaram_count == 5)
      {
      for(keypress_count=0;keypress_count<2;keypress_count++)
      {
       if (keypress_count==1)
       {
         lcd_cursorblink_flag=1;
       }
        key_press();
        if (keypress_count==1)
       {
         lcd_cursorblink_flag=0;
       }
       alaram_time_5_3[keypress_count]= glo_line;
      }
       alaram_time5_d = ((alaram_time_5_3[0]-0x30)*10) + (alaram_time_5_3[1]-0x30);
       printf("alaram time5_d is %d",alaram_time5_d);

      pagevalue=3;
      column=36;
      lcd_cursorblink_flag=1;
      lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      lcd_cursorblink_flag=0;
      keypress_count=0;
      for(keypress_count=0;keypress_count<2;keypress_count++)
      {
       if (keypress_count==1)
       {
         lcd_cursorblink_flag=1;
       }
        key_press();
```

```
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_5_4[keypress_count]= glo_line;
}
 alaram_time5_o = ((alaram_time_5_4[0]-0x30)*10) + (alaram_time_5_4[1]-0x30);
 printf("alaram time5_o is %d",alaram_time5_o);
pagevalue=3;
column=54;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
keypress_count=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
  key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_5_1[keypress_count]= glo_line;
}
 alaram_time5_h = ((alaram_time_5_1[0]-0x30)*10) + (alaram_time_5_1[1]-0x30);
 printf("alaram time5_h is %d",alaram_time5_h);
pagevalue=3;
column=72;
lcd_cursorblink_flag=1;
lcd_cursor_column_flag=1;
lcd_changelocation();
lcd_cursor_column_flag=0;
lcd_cursorblink_flag=0;
for(keypress_count=0;keypress_count<2;keypress_count++)
{
 if (keypress_count==1)
 {
   lcd_cursorblink_flag=1;
 }
 key_press();
  if (keypress_count==1)
 {
   lcd_cursorblink_flag=0;
 }
 alaram_time_5_2[keypress_count]= glo_line;
	}
	alaram_time5_m = ((alaram_time_5_2[0]-0x30)*10) + (alaram_time_5_2[1]-0x30);
	printf("alaram time5_m is %d",alaram_time5_m);

	 if( (alaram_time5_m > 60) || (alaram_time5_h > 23) || (alaram_time5_d > 31) ||
(alaram_time5_o > 12))
	{
	 lcd_color_red();
	 wait(1);
	 lcd_color_green();
	 goto    page3start;
	}

	}
	touchpressed_done_flag2=0;
	while(touchpressed_done_flag2==0)
	{
	 xcor=0;
	 ycor=0;
	 for(i=0;i<5;i++)
	 {
	   adc_read();
	   if(((xcor > 0x5200 && xcor< 0x7400)&&(ycor >0x7100 && ycor< 0x7C00)))
	   {
	    printf("\n\rsuccess\n\r");
	    touchpressed_done_flag2=1;
	    goto endloop1;
	   }
	}
	}
	endloop1:
	if(alaram_count == 1)
	{
	alaram1_lcd_flag=1;
	alaram1();
	}
	else if(alaram_count == 2)
	{
	alaram2_lcd_flag=1;
	alaram2();
	}
	else if(alaram_count == 3)
	{
	alaram3_lcd_flag=1;
	alaram3();
	}

	else if(alaram_count == 4)
	{
	alaram4_lcd_flag=1;
```

```
        alaram4();
        }

        else if(alaram_count == 5)
        {
        alaram5_lcd_flag=1;
        alaram5();
        }
}

void login()
{
   while(loginstatus != 1)
   {
   lcd_clear1(); //clear the lcd diplay without clearing the RTC on LCD
   login_write();
   login_user();

   }
}

void lcd_write_settime()
{
   lcd_write_character('S');//S
   lcd_write_character('E');//E
   lcd_write_character('T');//T
   lcd_write_character(0x00);//space
   lcd_write_character('T');//T
   lcd_write_character('I');//I
   lcd_write_character('M');//M
   lcd_write_character('E');//E
}

void lcd_write_timeformat()
{
   lcd_write_character('D');//D
   lcd_write_character('D');//D
   lcd_write_character('/');//:
   lcd_write_character('M');//M
   lcd_write_character('M');//M
   lcd_write_character(0x00);//space
   lcd_write_character('H');//H
   lcd_write_character('H');//H
   lcd_write_character(':');//:
   lcd_write_character('M');//M
   lcd_write_character('M');//M

}

void  lcd_write_reset()
```

```
{
   lcd_write_character(0x02);//Block
   lcd_write_character('R');//R
   lcd_write_character('E');//E
   lcd_write_character('S');//S
   lcd_write_character('E');//E
   lcd_write_character('T');//T
   lcd_write_character(0x02);//Block
}




void  lcd_write_describe()
{

   lcd_write_character('D');//D
   lcd_write_character('E');//E
   lcd_write_character('S');//S
   lcd_write_character('C');//C
   lcd_write_character('R');//R
   lcd_write_character('I');//I
   lcd_write_character('B');//B
   lcd_write_character('E');//E

}

void  lcd_write_done()
{
   lcd_write_character(0x02);//Block
   lcd_write_character('D');//D
   lcd_write_character('O');//O
   lcd_write_character('N');//N
   lcd_write_character('E');//E
   lcd_write_character(0x02);//Block
}

void  lcd_write_add()
{
   lcd_write_character(0x02);//Block
   lcd_write_character('A');//A
   lcd_write_character('D');//D
   lcd_write_character('D');//D
   lcd_write_character(0x02);//Block
}

void  lcd_write_delete()
{
   lcd_write_character(0x02);//Block
   lcd_write_character('D');//D
   lcd_write_character('E');//E
   lcd_write_character('L');//L
   lcd_write_character('E');//E
   lcd_write_character('T');//T
   lcd_write_character('E');//E
   lcd_write_character(0x02);//Block
}

void login_write()
{
   pagevalue = 0;
   column=60;
     lcd_cursor_column_flag=1;
     lcd_changelocation();
     lcd_cursor_column_flag=0;
   login_write_login();
   pagevalue = 2;
   column=00;
         lcd_cursor_column_flag=1;
     lcd_changelocation();
     lcd_cursor_column_flag=0;

   login_write_username();
   pagevalue = 4;
   column=00;
         lcd_cursor_column_flag=1;
     lcd_changelocation();
     lcd_cursor_column_flag=0;

   login_write_password();
}

void login_write_login()
{
   lcd_write_character('L');//l
   lcd_write_character('O');//o
   lcd_write_character('G');//g
   lcd_write_character('I');//i
   lcd_write_character('N');//n
}
void login_write_username()
{
   lcd_write_character('U');//u
   lcd_write_character('S');//s
   lcd_write_character('E');//e
   lcd_write_character('R');//r
   lcd_write_character('N');//n
   lcd_write_character('A');//a
```

```
    lcd_write_character('M');//m
    lcd_write_character('E');//e
    lcd_write_character(':');//:
}



void login_write_password()
{
    lcd_write_character('P');//p
    lcd_write_character('A');//a
    lcd_write_character('S');//s
    lcd_write_character('S');//s
    lcd_write_character('W');//w
    lcd_write_character('O');//o
    lcd_write_character('R');//r
    lcd_write_character('D');//d
    lcd_write_character(':');//:
}

void hide_character()
{
     column=column-6;
     lcd_changelocation();
     lcd_write_character('*');
}

void login_write_error()
{
    lcd_write_character('I');//i
    lcd_write_character('N');//n
    lcd_write_character('V');//v
    lcd_write_character('A');//a
    lcd_write_character('L');//l
    lcd_write_character('I');//i
    lcd_write_character('D');//d
    lcd_write_character(0x00);//space
    lcd_write_character('C');//c
    lcd_write_character('R');//r
    lcd_write_character('E');//e
    lcd_write_character('D');//d
    lcd_write_character('E');//e
    lcd_write_character('N');//n
    lcd_write_character('T');//t
    lcd_write_character('I');//i
    lcd_write_character('A');//a
    lcd_write_character('L');//l
    lcd_write_character('S');//s
}
```

```
void lcd_write_deletetask()
{
    lcd_write_character('D');//i
    lcd_write_character('E');//n
    lcd_write_character('L');//v
    lcd_write_character('E');//a
    lcd_write_character('T');//l
    lcd_write_character('E');//i
    lcd_write_character(0x00);//space
    lcd_write_character('T');//c
    lcd_write_character('A');//r
    lcd_write_character('S');//e
    lcd_write_character('K');//d
    lcd_write_character(':');//d
}

void login_user()
{

   unsigned char i;
   unsigned char username_enter[5];
   unsigned char password_enter[5];
   unsigned char count_username=0;
   unsigned char count_password=0;

    pagevalue = 2;
    column=60;
    lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;
      keypress_count=0;

    for(keypress_count=0;keypress_count<5;keypress_count++)
    {
       key_press();                                   //wait till the key is pressed
       username_enter[keypress_count]= glo_line;                //value of username is stored
in the following
    }
     pagevalue = 4;
     column=60;
     lcd_cursor_column_flag=1;
      lcd_changelocation();
      lcd_cursor_column_flag=0;

    for(keypress_count=0;keypress_count<5;keypress_count++)
    {
       key_press();
       wait(.3);
       lcd_timer_flag=1;
```

```
        hide_character();
        lcd_timer_flag=0;
        password_enter[keypress_count]= glo_line;
      }

      for(i=0;i<5;i++)
      {
         if(main_username[i] == username_enter[i])
         {
            count_username++;
         }
      }

      for(i=0;i<5;i++)
      {
         if(main_username[i] == password_enter[i])
         {
            count_password++;
         }
      }


      if((count_username == 5) && (count_password ==5))
       {
         loginstatus=1;
       }
       else
       {
         pagevalue = 5;
         column=6;
         lcd_cursor_column_flag=1;
         lcd_changelocation();
         lcd_cursor_column_flag=0;
         login_write_error();
         lcd_color_red();
         wait(1);
         lcd_color_green();
//       wait(1);
       }

   }

   void lcd_write_description()
   {
        unsigned char i;
        for(i=0;i<desc_count;i++)
        {
        lcd_write_character(alaram_des_1[i]);  //write the value of the description 1
        }
```

```
}

void lcd_write_description2()
{
    unsigned char i;

    for(i=0;i<desc_count2;i++)
    {
    lcd_write_character(alaram_des_2[i]);
    }

}

void lcd_write_description3()
{
    unsigned char i;

    for(i=0;i<desc_count3;i++)
    {
    lcd_write_character(alaram_des_3[i]);
    }

}

void lcd_write_description4()
{
    unsigned char i;
    for(i=0;i<desc_count4;i++)
    {
    lcd_write_character(alaram_des_4[i]);
    }

}

void lcd_write_description5()
{
    unsigned char i;
    for(i=0;i<desc_count5;i++)
    {
    lcd_write_character(alaram_des_5[i]);
    }

}

void lcd_write_time1()
{
    unsigned char i;
    for(i=0;i<2;i++)
    {
    lcd_write_character(alaram_time_1_3[i]);    //value of the month of alaram1
```

```
      }
      lcd_write_character('/');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_1_4[i]);  //value of the day of alaram1
      }
      lcd_write_character(0x00);
     for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_1_1[i]);  //value of the hour of alaram1
      }
      lcd_write_character(':');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_1_2[i]);  //value of the minute of alaram1
      }

}

//Put the value of time 2 on the LCD screen
void lcd_write_time2()
{
      unsigned char i;
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_2_3[i]);
      }
      lcd_write_character('/');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_2_4[i]);
      }
      lcd_write_character(0x00);
     for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_2_1[i]);
      }
      lcd_write_character(':');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_2_2[i]);
      }

}

//Put the value of time 3 on the LCD screen
void lcd_write_time3()
{
      unsigned char i;
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_3_3[i]);
      }
      lcd_write_character('/');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_3_4[i]);
      }
      lcd_write_character(0x00);
     for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_3_1[i]);
      }
      lcd_write_character(':');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_3_2[i]);
      }

}

//Put the value of time 4 on the LCD screen
void lcd_write_time4()
{
      unsigned char i;
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_4_3[i]);
      }
      lcd_write_character('/');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_4_4[i]);
      }
      lcd_write_character(0x00);
     for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_4_1[i]);
      }
      lcd_write_character(':');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_4_2[i]);
      }

}
```

```
//Put the value of time 5 on the LCD screen
void lcd_write_time5()
{
      unsigned char i;
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_5_3[i]);
      }
      lcd_write_character('/');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_5_4[i]);
      }
      lcd_write_character(0x00);
     for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_5_1[i]);
      }
      lcd_write_character(':');
      for(i=0;i<2;i++)
      {
      lcd_write_character(alaram_time_5_2[i]);
      }

}




/*------------------------------------------------------------------------------------------------------------
-------------------------------------------------------
 * Raghunath Reddy
 * ECEN 5613, Graphic LCD
 * Fall 2016,Prof. Mc Clure
 * University of Colorado at Boulder
 * ------------------------------------------
 * This file helps the user to contrrol TG12864H3-05A which has option for displaying
cursor, this library also contains functions which are application specific
 * -----------------------------------------------------------------------------------------------------------
----------------------------------------------------- */
```

```
/* Gloabalized variables----------------------------------------------------- */
//Did not inculude pointer to pointer decelat=ration so had to use 5 variables for 5 dfifferent
values assingned with timer and most of these variable are used side main function
int alaram1sec=0;                                   //Has the value of the alaram 1 in sec
int alaram2sec=0;
int alaram3sec=0;
int alaram4sec=0;
int alaram5sec=0;
unsigned char string_count;
unsigned char page[8] = {3,2,1,0,7,6,5,4};                  //page values are matched with
what can be exactly seen on LCD , for better and easy understanding
unsigned char column;                                //stores the current value of the column
unsigned char pagevalue;                              //Store the current valuee of the page
unsigned char glo_line;                              //Store the value of the character that is
entered on the keypad
unsigned char lcd_timer_flag=0;                           //Timer to be displayed based on
this flag
unsigned char lcd_cursor_column_flag=1;                         //Cursor to be displayed in the
next column based on this flag
unsigned char lcd_cursorblink_flag=0;                        //Based this flag cursor may or
may not be displayed
unsigned char alaram_count;                              //Counts the number of alarams set
unsigned char alaram1_occured_flag=0;                         //Indication if the alaram has
occured or not
unsigned char alaram2_occured_flag=0;
unsigned char alaram3_occured_flag=0;
unsigned char alaram4_occured_flag=0;
unsigned char alaram5_occured_flag=0;
unsigned char main_username[5]={'R','A','G','H','U'};              //Login credentials
unsigned char main_password[5]={'R','A','G','H','U'};              //Login credentials
unsigned char loginstatus;

unsigned char alaram_time1_h;                            //Contains the number hours
entered by the user for the alaram1
unsigned char alaram_time1_m;                             //Contains the number minutes
entered by the user for the alaram1
unsigned char alaram_time1_d;                            //Contains the number days
entered by the user for the alaram1
unsigned char alaram_time1_o;                            //Contains the number months
entered by the user for the alaram1
unsigned char lcd_refresh=0;

unsigned char alaram_time2_h;
unsigned char alaram_time2_m;
unsigned char alaram_time2_d;
unsigned char alaram_time2_o;

unsigned char alaram_time3_h;
unsigned char alaram_time3_m;
unsigned char alaram_time3_d;
unsigned char alaram_time3_o;

unsigned char alaram_time4_h;
unsigned char alaram_time4_m;
unsigned char alaram_time4_d;
unsigned char alaram_time4_o;

unsigned char alaram_time5_h;
unsigned char alaram_time5_m;
unsigned char alaram_time5_d;
unsigned char alaram_time5_o;

unsigned char alaram_des_1[20];                         //Stores the description
alloted to alaram1
unsigned char alaram_des_2[20];
unsigned char alaram_des_3[20];
unsigned char alaram_des_4[20];
unsigned char alaram_des_5[20];

unsigned char desc_count;                             //store the number of words
associated with the descriprion 1
unsigned char desc_count2;
unsigned char desc_count3;
unsigned char desc_count4;
unsigned char desc_count5;

unsigned char alaram1_lcd_flag=0;                       //Tells the controller to
display the alaram information on the LCD or not
unsigned char alaram2_lcd_flag=0;
unsigned char alaram3_lcd_flag=0;
unsigned char alaram4_lcd_flag=0;
unsigned char alaram5_lcd_flag=0;

unsigned char alaram_time_delete=0;
unsigned char touchpressed_done2_flag=0;                  //Flag to indicate whether
done button on touch screen is pressed or not
int nextpage;

unsigned char alaram1_flag;
unsigned char alaram2_flag;
unsigned char alaram3_flag;
unsigned char alaram4_flag;
unsigned char alaram5_flag;


unsigned char alaram_time_1_2[3];                      //Store the ascii values of the
hours enetered
unsigned char alaram_time_1_1[3];                      //Store the ascii values of the
minutes enetered
```

```
unsigned char alaram_time_1_4[3];                    //Store the ascii values of the
days enetered
unsigned char alaram_time_1_3[3];                    //Store the ascii values of the
months enetered

unsigned char alaram_time_2_2[3];
unsigned char alaram_time_2_1[3];
unsigned char alaram_time_2_4[3];
unsigned char alaram_time_2_3[3];

unsigned char alaram_time_3_2[3];
unsigned char alaram_time_3_1[3];
unsigned char alaram_time_3_4[3];
unsigned char alaram_time_3_3[3];

unsigned char alaram_time_4_2[3];
unsigned char alaram_time_4_1[3];
unsigned char alaram_time_4_4[3];
unsigned char alaram_time_4_3[3];

unsigned char alaram_time_5_2[3];
unsigned char alaram_time_5_1[3];
unsigned char alaram_time_5_4[3];
unsigned char alaram_time_5_3[3];

unsigned char touchpressed_add_flag=0;                        //indicates the status
whether add button on the touchscreen is pressed or not
unsigned char touchpressed_delete_flag=0;                      //indicates the status
whether add button on the touchscreen is pressed or not



unsigned char lcdbuffer[1530]=          //BITMAP OF THE LCD inclues A-Z 0-9 and few
special characters used in the application
{
0x00,0x00,0x00,0x00,0x00,0x00,//0x00
0x00,0x01,0x01,0x01,0x01,0x01,//0x01
0x00,0xFF,0xFF,0xFF,0xFF,0xFF,//0x02
0x00,0x00,0x00,0x00,0x00,0x00,//0x03
0x00,0x00,0x00,0x00,0x00,0x00,//0x04
0x00,0x00,0x00,0x00,0x00,0x00,//0x05
0x00,0x00,0x00,0x00,0x00,0x00,//0x06
0x00,0x00,0x00,0x00,0x00,0x00,//0x07
0x00,0x00,0x00,0x00,0x00,0x00,//0x08
0x00,0x00,0x00,0x00,0x00,0x00,//0x09
0x00,0x00,0x00,0x00,0x00,0x00,//0x0A
0x00,0x00,0x00,0x00,0x00,0x00,//0x0B
0x00,0x00,0x00,0x00,0x00,0x00,//0x0C
0x00,0x00,0x00,0x00,0x00,0x00,//0x0D
0x00,0x00,0x00,0x00,0x00,0x00,//0x0E
```

```
0x00,0x00,0x00,0x00,0x00,0x00,//0x0F
0x00,0x00,0x00,0x00,0x00,0x00,//0x10
0x00,0x00,0x00,0x00,0x00,0x00,//0x11
0x00,0x00,0x00,0x00,0x00,0x00,//0x12
0x00,0x00,0x00,0x00,0x00,0x00,//0x13
0x00,0x00,0x00,0x00,0x00,0x00,//0x14
0x00,0x00,0x00,0x00,0x00,0x00,//0x15
0x00,0x00,0x00,0x00,0x00,0x00,//0x16
0x00,0x00,0x00,0x00,0x00,0x00,//0x17
0x00,0x00,0x00,0x00,0x00,0x00,//0x18
0x00,0x00,0x00,0x00,0x00,0x00,//0x19
0x00,0x00,0x00,0x00,0x00,0x00,//0x1A
0x00,0x00,0x00,0x00,0x00,0x00,//0x1B
0x00,0x00,0x00,0x00,0x00,0x00,//0x1C
0x00,0x00,0x00,0x00,0x00,0x00,//0x1D
0x00,0x00,0x00,0x00,0x00,0x00,//0x1E
0x00,0x00,0x00,0x00,0x00,0x00,//0x1F
0x00,0x00,0x00,0x00,0x00,0x00,//0x20
0x00,0x00,0x00,0x00,0x00,0x00,//0x21
0x00,0x00,0x00,0x00,0x00,0x00,//0x22
0x00,0x00,0x00,0x00,0x00,0x00,//0x23
0x00,0x00,0x00,0x00,0x00,0x00,//0x24
0x00,0x00,0x00,0x00,0x00,0x00,//0x25
0x00,0x00,0x00,0x00,0x00,0x00,//0x26
0x00,0x00,0x00,0x00,0x00,0x00,//0x27
0x00,0x00,0x00,0x00,0x00,0x00,//0x28
0x00,0x00,0x00,0x00,0x00,0x00,//0x29
0x00,0x22,0x14,0x3C,0x14,0x22,//0x2A
0x00,0x00,0x00,0x00,0x00,0x00,//0x2B
0x00,0x00,0x00,0x00,0x00,0x00,//0x2C
0x00,0x00,0x00,0x00,0x00,0x00,//0x2D
0x00,0x00,0x00,0x00,0x00,0x00,//0x2E
0x00,0x20,0x10,0x08,0x04,0x02,//0x2F--/
0x00,0x3e,0x45,0x49,0x51,0x3e,//0x30--0
0x00,0x00,0x11,0x31,0x7F,0x01,//0x31--1
0x00,0x21,0x43,0x45,0x49,0x31,//0x32--2
0x00,0x42,0x41,0x51,0x69,0x46,//0x33--3
0x00,0x0c,0x14,0x24,0x7F,0x04,//0x34--4
0x00,0x71,0x51,0x51,0x51,0x4E,//0x35--5
0x00,0x3E,0x49,0x49,0x49,0x06,//0x36--6
0x00,0x40,0x47,0x48,0x50,0x60,//0x37--7
0x00,0x36,0x49,0x49,0x49,0x36 ,//0x38--8
0x00,0x30,0x49,0x49,0x49,0x3E ,//0x39--9
0x00,0x00,0x00,0x24,0x00,0x00,//0x3A--:-37
0x00,0x00,0x00,0x00,0x00,0x00,//0x3B
0x00,0x00,0x00,0x00,0x00,0x00,//0x3C
0x00,0x00,0x00,0x00,0x00,0x00,//0x3D
0x00,0x00,0x00,0x00,0x00,0x00,//0x3E
0x00,0x00,0x00,0x00,0x00,0x00,//0x3F
0x00,0x00,0x00,0x00,0x00,0x00,//0x40
```

```
 0x00,0x3f,0x44,0x44,0x44,0x3F,    //A-0x41
0x00,0x7f,0x49,0x49,0x49,0x36,   //B-0x42
 0x00,0x3e,0x41,0x41,0x41,0x22,    //C-0x43
 0x00,0x7F,0x41,0x41,0x22,0x1c,    //D-0x44
 0x00,0x7F,0x49,0x49,0x49,0x41,    //E-0x45
 0x00,0x7F,0x48,0x48,0x48,0x40,   //F-0x46
 0x00,0x3e,0x41,0x49,0x49,0x2e,  //G-0x47
 0x00,0x7f,0x08,0x08,0x08,0x7f ,  //H-0x48
 0x00,0x41,0x41,0x7F,0x41,0x41,   //I-0x49
 0x00,0x02,0x41,0x41,0x7e,0x40 ,   //J-0x4a
 0x00,0x7f,0x08,0x14,0x22,0x41,    //K-0x4b
 0x00,0x7F,0x01,0x01,0x01,0x01,    //L-0x4c
  0x00,0x7f,0x20,0x18,0x20,0x7f,   //M-0x4d
  0x00,0x7f,0x10,0x08,0x04,0x7f,  //N-0x4e
  0x00,0x3e,0x41,0x41,0x41,0x3e,   //O-0x4f
  0x00,0x7f,0x48,0x48,0x48,0x30,   //P-0x50
  0x00,0x3e,0x41,0x45,0x42,0x3f,  //Q-0x51
  0x00,0x7f,0x48,0x4c,0x4a,0x31,   //R-0x52
  0x00,0x31,0x49,0x49,0x49,0x46,   //S-0x53
  0x00,0x40,0x40,0x7f,0x40,0x40,  //T-0x54
  0x00,0x7e,0x01,0x01,0x01,0x7e,  //U-0x55
  0x00,0x7C,0x02,0x01,0x02,0x7C,   //V-0x56
  0x00,0x7e,0x01,0x0e,0x01,0x7e,   //W-0x57
  0x00,0x63,0x14,0x08,0x14,0x63,   //X-0x58
  0x00,0x70,0x08,0x07,0x08,0x70,   //Y-0x59
  0x00,0x43,0x45,0x49,0x51,0x61,   //Z-0x5A
  0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,
  0x00,0x00,0x00,0x00,0x00,0x00,

  0x00,0x00,0x00,0x00,0x00,0x00,   //a
  0x00,0x00,0x00,0x00,0x00,0x00,   //b
  0x00,0x3e,0x41,0x41,0x41,0x22,   //c
  0x00,0x00,0x00,0x00,0x00,0x00,   //d
  0x00,0x7F,0x49,0x49,0x49,0x41,   //e
  0x00,0x00,0x00,0x00,0x00,0x00,   //f
  0x00,0x00,0x00,0x00,0x00,0x00,   //g
  0x00,0x00,0x00,0x00,0x00,0x00,   //h
  0x00,0x00,0x00,0x00,0x00,0x00,   //i
  0x00,0x00,0x00,0x00,0x00,0x00,   //j
  0x00,0x00,0x00,0x00,0x00,0x00,   //k
  0x00,0x00,0x00,0x00,0x00,0x00,   //l
  0x00,0x00,0x00,0x00,0x00,0x00,   //m
  0x00,0x00,0x00,0x00,0x00,0x00,   //n
  0x00,0x3e,0x41,0x41,0x41,0x3e,   //o
  0x00,0x00,0x00,0x00,0x00,0x00,   //p
  0x00,0x00,0x00,0x00,0x00,0x00,   //q
  0x00,0x00,0x00,0x00,0x00,0x00,   //r
  0x00,0x00,0x00,0x00,0x00,0x00,   //s
  0x00,0x00,0x00,0x00,0x00,0x00,   //t

  0x00,0x00,0x00,0x00,0x00,0x00,   //u
   0x00,0x7C,0x02,0x01,0x02,0x7C,   //v
  0x00,0x00,0x00,0x00,0x00,0x00,   //w
  0x00,0x00,0x00,0x00,0x00,0x00,   //x
  0x00,0x00,0x00,0x00,0x00,0x00,   //y
  0x00,0x00,0x00,0x00,0x00,0x00   //z

};



/* Prototypes --------------------------------------------------- */
 /*-----------------------------------------------------------------------------------------------------------
-------------------------------------------------------
  * Initialize_lcd_pinouts()
  * Purpose:     Initializing the LCD pins
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void initialize_lcd_pinouts();

 /*-----------------------------------------------------------------------------------------------------------
-------------------------------------------------------
  * Initialize_lcd_pinouts()
  * Purpose:     Initializing the LCD and power supply to the LCD
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void initialize_lcd();

/*------------------------------------------------------------------------------------------------------------
------------------------------------------------------
  * lcd_cmd()
  * Purpose:     To send a command to the LCD
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void lcd_cmd(unsigned char);

/*------------------------------------------------------------------------------------------------------------
------------------------------------------------------
  * lcd_data()
  * Purpose:     To send a data to the LCD by making A0 high
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
```

```
void lcd_data(unsigned char);

/*----------------------------------------------------------------------------------------------------
-----------------------------------------------------
  * dealy_us_lcd()
  * Purpose:     A particular delay set after several trial and error methods
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void delay_us_lcd();

/*----------------------------------------------------------------------------------------------------
-----------------------------------------------------
  * lcd_clear()
  * Purpose:     Clears the entire LCD
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void lcd_clear();

/*----------------------------------------------------------------------------------------------------
-----------------------------------------------------
  * lcd_write_character()
  * Purpose:     write the character on to the LCDbased on the input provided expexting ascii
values in that
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void lcd_write_character(unsigned char line);


/*----------------------------------------------------------------------------------------------------
-----------------------------------------------------
  * cursor_blink()
  * Purpose:     Special character included for the cursor display as it not part of the LCD
hardware
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void cursor_blink();

/*----------------------------------------------------------------------------------------------------
-----------------------------------------------------
  * column_change()
  * Purpose:     Changes the column in which lcd is currently in by looking at the value of
column
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
void column_change();
```

```
/*----------------------------------------------------------------------------------------------------
--------------------------------------------------
  * check_column()
  * Purpose:     Checks if the LCD as reached the end of th column if yes changes it to the
next row/Page
  * Calcuations: None
  * Return:      None
  *---------------------------------------------------------- */
void check_column();

/*----------------------------------------------------------------------------------------------------
--------------------------------------------------
  * lcd_clear1()
  * Purpose:     Clears the LCD but not the RTC
  * Calcuations: None
  * Return:      None
  *---------------------------------------------------------- */
void lcd_clear1();

/*----------------------------------------------------------------------------------------------------
--------------------------------------------------
  * lcd_changelocation()
  * Purpose:     changes the value of the location of lcd by using the page value and column
function
  * Calcuations: None
  * Return:      None
  *---------------------------------------------------------- */
void lcd_changelocation();

/*----------------------------------------------------------------------------------------------------
--------------------------------------------------
  * lcd_write_character()
  * Purpose:     write the character on to the LCDbased on the input provided expexting ascii
values in that
  * Calcuations: None
  * Return:      None
  *---------------------------------------------------------- */
void lcd_write_string(unsigned char);

/*----------------------------------------------------------------------------------------------------
--------------------------------------------------
  * lcd_color_red()
  * Purpose:     Changes the background color to red
  * Calcuations: None
  * Return:      None
  *---------------------------------------------------------- */
void lcd_color_red();
```

```
/*----------------------------------------------------------------------------------------------------
------------------------------------------------------
  * lcd_color_green()
  * Purpose:     Changes the background color of LCD to green
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------ */
void lcd_color_green();

/*----------------------------------------------------------------------------------------------------
------------------------------------------------------
  * lcd_color_green()
  * Purpose:     Changes the background color of LCD to red
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------ */
void lcd_color_blue();

/*----------------------------------------------------------------------------------------------------
------------------------------------------------------
  * alaram1()
  * Purpose:     Sets the value of the alaram by taking the user inputs for the page3
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------ */
void alaram1();

/*----------------------------------------------------------------------------------------------------
------------------------------------------------------
  * alaram2()
  * Purpose:     Sets the value of the alaram by taking the user inputs for the page3
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------ */
void alaram2();

/*----------------------------------------------------------------------------------------------------
------------------------------------------------------
  * alaram3()
  * Purpose:     Sets the value of the alaram by taking the user inputs for the page3
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------ */
void alaram3();

/*----------------------------------------------------------------------------------------------------
------------------------------------------------------
  * alaram4()
  * Purpose:     Sets the value of the alaram by taking the user inputs for the page3
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------------ */
void alaram4();




void lcd_color_red()
{
  LPC_GPIO2 -> FIOCLR |= (1<<0);
  LPC_GPIO2 -> FIOSET |= (1<<1);
  LPC_GPIO2 -> FIOSET |= (1<<2);

}

void lcd_color_blue()
{
  LPC_GPIO2 -> FIOSET |= (1<<0);
  LPC_GPIO2 -> FIOSET |= (1<<1);
  LPC_GPIO2 -> FIOCLR |= (1<<2);

}

void lcd_color_green()
{
  LPC_GPIO2 -> FIOSET |= (1<<0);
  LPC_GPIO2 -> FIOCLR |= (1<<1);
  LPC_GPIO2 -> FIOSET |= (1<<2);

}


void lcd_changelocation()
{
    unsigned char i;
    if(lcd_cursorblink_flag==0)                              //To check the corner conditions as in
cursor blink should occur during the column change
    {
    if(lcd_cursor_column_flag==1)
    {
          for(i=0;i<6;i++)
    {
     lcd_data(lcdbuffer[i+(6*0x00)]);                      //as there is a column change make sure
the previous cursor is erased and moved to the new location
    }
    }
    }
        lcd_cmd(0xB0 | page[pagevalue]);
        column_change();                              //function to change the column
   if(lcd_cursor_column_flag==1)
   {
       cursor_blink();                         //display the cursor in new position
   }
}

void lcd_write_string(unsigned char *lcdstring)
{
   unsigned char i;
   for(i=0;i<string_count;i++)
   {
      lcd_write_character(lcdstring[i]);           //to write the value of string on to the lcd by
using the lcd_write_character function
   }

}

void lcd_write_character(unsigned char line)
{
 if(lcd_timer_flag==0)
 {
 glo_line=line;
 }
 unsigned char i;
 for(i=0;i<6;i++)
 {
    //lcd_data(0xff);
    lcd_data(lcdbuffer[i+(6*line)]);                              //use lcddata to write one
column at a time
    column++;                                              // keep track of the column
 }
    check_column();
    if(lcd_cursorblink_flag == 0)
    {
    cursor_blink();
    }

}

void check_column()
{

   if(column==126)                           //checking the end of the column
   {
         if(pagevalue == 7)                     //checking the end of the pages
          {
           pagevalue = 0;
          }
         else
         {
            pagevalue=pagevalue+1;
            }
        lcd_cmd(0xB0 | page[pagevalue]);
        column=0x00;
        column_change();

    }


}



void initialize_lcd()
{

  lcd_color_green();
  LPC_GPIO0 -> FIOCLR |= (1<<4);              //Reset condition--L
  wait(.5);                                   //500ms suggested by the datasheet for initializing the system
  LPC_GPIO0 -> FIOSET |= (1<<4);              //Reset condition--H
  lcd_cmd(0xA3);                              //LCD bias 1/7th
  wait(0.1);                                  //100msec delay
  lcd_cmd(0xA0);                              //ADC select 0-normal
  wait(0.1);
  lcd_cmd(0xC0);                              //SHL select 0-normal direction
  wait(0.1);
  lcd_cmd(0x40);                              //Initial display line
  wait(0.1);
  lcd_cmd(0x2C);                              //voltage regulator
  wait(.5);
  lcd_cmd(0x2e);                              //voltage regulator
  wait(.5);
  lcd_cmd(0x2f);                              //voltage regulator
  wait(.5);
  lcd_clear();
  lcd_cmd(0xAF);                              //Display ON
  wait(.2);
  pagevalue=2;
  column=0x78;
  lcd_changelocation();

}

void column_change()
{
    unsigned char columnh;
    unsigned char columnl;
    columnh= column & 0xf0;
    columnh= columnh >> 4;
    columnl= column & 0x0f;
    lcd_cmd( 0x10 | columnh);
    lcd_cmd( 0x00 | columnl);

}

void initialize_lcd_pinouts()
{
    LPC_PINCON -> PINSEL3 &= ~((1<<5)|(1<<4));
    LPC_GPIO1-> FIODIR |= (1<<18);

    LPC_PINCON -> PINSEL0&= ~((1<<9)|(1<<8));                // /RST
    LPC_PINCON -> PINSEL0&= ~((1<<11)|(1<<10));              // A0
    LPC_PINCON -> PINSEL0&= ~((1<<20)|(1<<21));              //SID
    LPC_PINCON -> PINSEL0&= ~((1<<23)|(1<<22));              //SCLK
    LPC_GPIO0-> FIODIR |= (1<<4);                        //output
    LPC_GPIO0-> FIODIR |= (1<<5);                        //output
    LPC_GPIO0-> FIODIR |= (1<<10);                        //output
    LPC_GPIO0-> FIODIR |= (1<<11);                        //output


    //RGB outputs for the LCD SCREEN
    LPC_GPIO2-> FIODIR |= (1<<0);//output                    //R
    LPC_GPIO2-> FIODIR |= (1<<1);//output                    //G
    LPC_GPIO2-> FIODIR |= (1<<2);//output                    //B


}


//Bit banging to that data can be send to the LCD on the SDI line on the low to high transition
of sclk
void lcd_cmd (unsigned char cmdvalue)
{
    unsigned char k;
    unsigned char i;
    unsigned char j;
    k = 0x80;
    LPC_GPIO0 -> FIOCLR |= (1<<5);      //making A0 low as we are sending command
    for (i=0;i<=7;i++)
    {
    j = (cmdvalue & k);

    LPC_GPIO0 -> FIOCLR |= (1<<11);      //clearing the clock
    delay_us_lcd();                  //calculated delay for efficent operation of LCD
    if(j==0)
    {
    LPC_GPIO0 -> FIOCLR |= (1<<10);      //SID line cleared
    }
    else
    {
        LPC_GPIO0 -> FIOSET |= (1<<10);   //SID line high
    }
    delay_us_lcd();
    LPC_GPIO0 -> FIOSET |= (1<<11);      //setting the clock
    delay_us_lcd();
    k = k >> 1;
    }
}

//same procedure as lcd_cmd expect for A0 line is made high
void lcd_data (unsigned char datavalue)
{
    unsigned char k;
    unsigned char i;
    bool j;
    k = 0x80;
    LPC_GPIO0 -> FIOSET |= (1<<5);                                        //A0 is set
    for (i=0;i<=7;i++)
    {
    j = (datavalue & k);
    LPC_GPIO0 -> FIOCLR |= (1<<11);
    delay_us_lcd();
    if(j==0)
    {
        LPC_GPIO0 -> FIOCLR |= (1<<10);
    }
    else
    {
        LPC_GPIO0 -> FIOSET |= (1<<10);
    }
    delay_us_lcd();
    LPC_GPIO0 -> FIOSET |= (1<<11);
    delay_us_lcd();
    k = k >> 1;
    }
}

void delay_us_lcd()
{
    unsigned char i;
    for(i=0;i<=1;i++)
    {
    }

}
void lcd_clear()
{
  unsigned char p, c;
  unsigned char temp;
  temp=0xB0;
  for(p = 0; p < 8; p++)
  {

    lcd_cmd(temp);
    lcd_cmd(0x10);
    lcd_cmd(0x00);
    for(c = 0; c < 130; c++)
    {
      lcd_data(0x0);                                    //writting zero to all columns in all pages
    }
    temp=temp+1;

  }

}



//similar to lcd expext that rtc timer is not cleared
void lcd_clear1()
{
  unsigned char p, c;
  unsigned char temp;
  temp=0xB0;
  for(p = 0; p < 8; p++)
  {
    if(temp !=  0xB4)
    {
    lcd_cmd(temp);
    lcd_cmd(0x10);
    lcd_cmd(0x00);
    for(c = 0; c < 127; c++)
    {
      lcd_data(0x0);
    }

    }
    else
    {
    lcd_cmd(temp);
    lcd_cmd(0x10);
    lcd_cmd(0x00);
    for(c = 0; c <42 ; c++)
    {
      lcd_data(0x0);                       //RTC not cleared other than RTC others are cleared
    }

    }
    temp=temp+1;
  }
```

```
}



//TEST CODE FOR Troubelshooting
void lcd_test()
{
      lcd_write_character(0);
 lcd_write_character(0);
  lcd_write_character(1);
  lcd_write_character(1);
  lcd_write_character(2);
  lcd_write_character(3);
  lcd_write_character(4);
   lcd_write_character(5);
  lcd_write_character(6);
  lcd_write_character(7);
  lcd_write_character(8);
   lcd_write_character(9);
   lcd_cmd(0xB2);
   lcd_cmd(0x10);
   lcd_cmd(0x00);
   lcd_write_character(10);
   lcd_write_character(11);
   lcd_write_character(12);
   lcd_write_character(13);
   lcd_write_character(14);
   lcd_write_character(15);
   lcd_write_character(16);
   lcd_write_character(17);
   lcd_write_character(18);
   lcd_write_character(19);
   lcd_cmd(0xB1);
   lcd_cmd(0x10);
   lcd_cmd(0x00);
   lcd_write_character(20);
   lcd_write_character(21);
   lcd_write_character(22);
   lcd_write_character(23);
   lcd_write_character(24);
   lcd_write_character(25);
   lcd_write_character(26);
   lcd_write_character(27);
   lcd_write_character(28);
   lcd_write_character(29);
   lcd_write_character(30);
   lcd_write_character(31);
   lcd_write_character(32);
   lcd_write_character(33);
   lcd_write_character(34);
   lcd_write_character(35);
}
void cursor_blink()
{

  lcd_cmd(0xE0);                   //stop column address from incrementing
   unsigned char i;
   for(i=0;i<6;i++)
    {
     //lcd_data(0xff);
     lcd_data(lcdbuffer[i+(6*0x01)]);
    }
  lcd_cmd(0xEE);                  //start column address from incrementinf

}

// This is one very huge function with many conditional statements included and this function
every 60 seconds by using  repeteive interrupt
//as my declarations did not use pointer there are five similar sets of code
void lcd_time_display()
{
    char *q;
    unsigned char temp1;
    unsigned char temp2;
    unsigned char i;
    unsigned char lcd_touch_alaramout;
    unsigned char j;
    unsigned char k;
    unsigned char temp;
    lcd_refresh=1;
    temp1=column;
    temp2=pagevalue;
    column=42;
    pagevalue=7;
   lcd_cmd(0xB0|page[pagevalue]);
    column_change();
    i=0;
    time_t seconds = time(NULL);                // This function is taken from mbed library
    q=ctime(&seconds);                       //current value of time is stored
    while( *q != 0x00 )
    {
     temp=*q;
     timevalue[i]=temp;                    // read the time value in tis format 2015 nov 29  17:21:22
     q++;
     i++;
    }

     i=3;
    for(i=3;i<16;i++)
    {
      lcd_timer_flag=1;
      lcd_write_character(timevalue[i]);    //display only nov 29 12:21 format on LCD
    }
    i=0;

//    time_t seconds = time(NULL);
    lcd_timer_flag=0;
    column=temp1;
    pagevalue=temp2;
    lcd_cursor_column_flag=1;
    lcd_changelocation();
    lcd_cursor_column_flag=0;
          if ((seconds == (alaram1sec+2)) || (seconds == (alaram2sec+2)) || (seconds ==
(alaram3sec+2)) || (seconds == (alaram4sec+2)) || (seconds == (alaram5sec+2)))
      {
       //Check if any alaram occured

        if (seconds == (alaram1sec+2)) //check which alaram occured
        {
           alaram1_occured_flag =1;
        }
        if (seconds == (alaram2sec+2))
        {
           alaram2_occured_flag =1;
        }
        if (seconds == (alaram3sec+2))
        {
           alaram3_occured_flag =1;
        }
        if (seconds == (alaram4sec+2))
        {
           alaram4_occured_flag =1;
        }
        if (seconds == (alaram5sec+2))
        {
           alaram5_occured_flag =1;
        }

        lcd_cmd(0xAE);                              //Display OFF
        wait(.1);
        xcor=0;
        ycor=0;
        lcd_touch_alaramout=0;
        while(lcd_touch_alaramout==0)               //Wait till touch screen is pressed at any plce
        {
           LPC_GPIO2 -> FIOSET |= (1<<4);
           lcd_color_red();                     //Flashing of LCD background color
           for(j=0;j<5;j++)
           {
```

```
      for(k=0;k<3;k++)
       {
         for(i=0;i<5;i++)
           {
             adc_read();                 //read the value of adc connected to touchscreen
           }
    if(((xcor > 0x1000 )&&(ycor >0x1000)))
   {
    lcd_touch_alaramout=1;
    goto endalaram;
   }
 }

    }
    //printf("\n\r i am out of red \r\n");
    LPC_GPIO2 -> FIOCLR |= (1<<4);
    lcd_color_blue();
    for(j=0;j<5;j++)
    {
      for(k=0;k<3;k++)
       {
         for(i=0;i<5;i++)
           {
             adc_read();
           }
    if(((xcor > 0x1000 )&&(ycor >0x7100)))
   {
    printf("\n\rsuccess\n\r");
    //touchpressed_done_flag=1;
    lcd_touch_alaramout=1;
    goto endalaram;
    }
    }


    }
    //printf("\n\r i am out of blue \r\n");
    //LPC_GPIO2 -> FIOSET |= (1<<4);
    lcd_color_green();
    for(j=0;j<5;j++)
    {
      for(k=0;k<3;k++)
       {
         for(i=0;i<5;i++)
           {
             adc_read();
           }
    if(((xcor > 0x1000 )&&(ycor >0x7100)))
   {
    printf("\n\rsuccess\n\r");
```

```
        //touchpressed_done_flag=1;
        lcd_touch_alaramout=1;
        goto endalaram;
       }
          }

        }
       // printf("\n\r i am out of green \r\n");
     }
    }
    if(lcd_touch_alaramout==1)                    //Once the alaram is done it is important to
clear it and trafer the previous alaram to its position
    {
     endalaram:
     LPC_GPIO2 -> FIOCLR |= (1<<4);
     lcd_color_green();
      lcd_cmd(0xAF);                        //Display ON
       wait(.2);
     printf("\r\niam out\r\n");
     lcd_touch_alaramout=0;

     if(alaram1_occured_flag==1)
     {

        if(alaram_count == 1)
        {
         alaram1_lcd_flag = 0;            // make the flag for alarm 1 zer0
         alaram1sec=0;                  // make the alaram1 zero
         alaram_count--;                //decreement alarm count
        }
       if(alaram_count == 2)
       {
        desc_count=desc_count2;
        for(i=0;i<desc_count2;i++)
        {
        alaram_des_1[i]=alaram_des_2[i];    //exchange the descriptions between 1 and 2
asciivalue
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_1[i]=alaram_time_2_1[i];  //exchange the alaram between 1 and 2
hours  ascii value
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_2[i]=alaram_time_2_2[i];  //exchange the alaram between 1 and 2
minutes  asciivalue
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_3[i]=alaram_time_2_3[i];  //exchange the alaram between 1 and 2
Months asciivalue
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_4[i]=alaram_time_2_4[i];  //exchange the alaram between 1 and 2
Day ascii value
        }
        alaram_time1_h=alaram_time2_h;      //decimal values of hour exchanged between
alaram 1 and alaram2
        alaram_time1_m=alaram_time2_m;      //decimal values of minutes exchanged
between alaram 1 and alaram2
        alaram_time1_d=alaram_time2_d;      //decimal values of days exchanged between
alaram 1 and alaram2
        alaram_time1_o=alaram_time2_o;      //decimal values of months exchanged between
alaram 1 and alaram2
        alaram_count--;
        alaram2sec=0;
        alaram2_lcd_flag = 0;
        alaram1();                          //reload the value of alaram1
       }
        if(alaram_count == 3)
       {
        desc_count=desc_count2;
        for(i=0;i<desc_count2;i++)
        {
        alaram_des_1[i]=alaram_des_2[i];
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_1[i]=alaram_time_2_1[i];
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_2[i]=alaram_time_2_2[i];
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_3[i]=alaram_time_2_3[i];
        }
        for(i=0;i<2;i++)
        {
         alaram_time_1_4[i]=alaram_time_2_4[i];
        }
        alaram_time1_h=alaram_time2_h;
        alaram_time1_m=alaram_time2_m;
        alaram_time1_d=alaram_time2_d;
        alaram_time1_o=alaram_time2_o;
```

```
 desc_count2=desc_count3;
 for(i=0;i<desc_count3;i++)
 {
 alaram_des_2[i]=alaram_des_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_1[i]=alaram_time_3_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_2[i]=alaram_time_3_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_3[i]=alaram_time_3_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_4[i]=alaram_time_3_4[i];
 }
 alaram_time2_h=alaram_time3_h;
 alaram_time2_m=alaram_time3_m;
 alaram_time2_d=alaram_time3_d;
 alaram_time2_o=alaram_time3_o;
 alaram_count--;
 alaram3sec=0;
 alaram1();
 alaram2();
 alaram3_lcd_flag = 0;
}


 if(alaram_count == 4)
{
 desc_count=desc_count2;
 for(i=0;i<desc_count2;i++)
 {
 alaram_des_1[i]=alaram_des_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_1[i]=alaram_time_2_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_2[i]=alaram_time_2_2[i];
 }
 for(i=0;i<2;i++)
 {
```

```
 alaram_time_1_3[i]=alaram_time_2_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_1_4[i]=alaram_time_2_4[i];
}
alaram_time1_h=alaram_time2_h;
alaram_time1_m=alaram_time2_m;
alaram_time1_d=alaram_time2_d;
alaram_time1_o=alaram_time2_o;


desc_count2=desc_count3;
for(i=0;i<desc_count3;i++)
{
alaram_des_2[i]=alaram_des_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_1[i]=alaram_time_3_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_2[i]=alaram_time_3_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_3[i]=alaram_time_3_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_4[i]=alaram_time_3_4[i];
}
alaram_time2_h=alaram_time3_h;
alaram_time2_m=alaram_time3_m;
alaram_time2_d=alaram_time3_d;
alaram_time2_o=alaram_time3_o;


desc_count3=desc_count4;
for(i=0;i<desc_count4;i++)
{
alaram_des_3[i]=alaram_des_4[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_1[i]=alaram_time_4_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_2[i]=alaram_time_4_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_3[i]=alaram_time_4_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_4[i]=alaram_time_4_4[i];
 }
 alaram_time3_h=alaram_time4_h;
 alaram_time3_m=alaram_time4_m;
 alaram_time3_d=alaram_time4_d;
 alaram_time3_o=alaram_time4_o;
 alaram_count--;
 alaram4sec=0;
 alaram1();
 alaram2();
 alaram3();
 alaram4_lcd_flag = 0;
}

 if(alaram_count == 5)
{
 desc_count=desc_count2;
 for(i=0;i<desc_count2;i++)
 {
 alaram_des_1[i]=alaram_des_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_1[i]=alaram_time_2_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_2[i]=alaram_time_2_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_3[i]=alaram_time_2_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_1_4[i]=alaram_time_2_4[i];
 }
 alaram_time1_h=alaram_time2_h;
 alaram_time1_m=alaram_time2_m;
 alaram_time1_d=alaram_time2_d;
 alaram_time1_o=alaram_time2_o;
```

```
desc_count2=desc_count3;
for(i=0;i<desc_count3;i++)
{
alaram_des_2[i]=alaram_des_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_1[i]=alaram_time_3_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_2[i]=alaram_time_3_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_3[i]=alaram_time_3_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_4[i]=alaram_time_3_4[i];
}
alaram_time2_h=alaram_time3_h;
alaram_time2_m=alaram_time3_m;
alaram_time2_d=alaram_time3_d;
alaram_time2_o=alaram_time3_o;


desc_count3=desc_count4;
for(i=0;i<desc_count4;i++)
{
alaram_des_3[i]=alaram_des_4[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_1[i]=alaram_time_4_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_2[i]=alaram_time_4_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_3[i]=alaram_time_4_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_4[i]=alaram_time_4_4[i];
}
alaram_time3_h=alaram_time4_h;
```

```
     alaram_time3_m=alaram_time4_m;
     alaram_time3_d=alaram_time4_d;
     alaram_time3_o=alaram_time4_o;


     desc_count4=desc_count5;
     for(i=0;i<desc_count5;i++)
     {
     alaram_des_4[i]=alaram_des_5[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_1[i]=alaram_time_5_1[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_2[i]=alaram_time_5_2[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_3[i]=alaram_time_5_3[i];
     }
     for(i=0;i<2;i++)
     {
      alaram_time_4_4[i]=alaram_time_5_4[i];
     }
     alaram_time4_h=alaram_time5_h;
     alaram_time4_m=alaram_time5_m;
     alaram_time4_d=alaram_time5_d;
     alaram_time4_o=alaram_time5_o;



     alaram_count--;
     alaram5sec=0;



     alaram1();
     alaram2();
     alaram3();
     alaram4();
     alaram5_lcd_flag = 0;
    }



}
```

```
if(alaram2_occured_flag==1)
{
    if(alaram_count == 2)
    {
     alaram_count--;
     alaram2sec=0;
     alaram2_lcd_flag = 0;
    // alaram1();
    }
     if(alaram_count == 3)
    {

     desc_count2=desc_count3;
     for(i=0;i<desc_count3;i++)
      {
     alaram_des_2[i]=alaram_des_3[i];
      }
     for(i=0;i<2;i++)
      {
      alaram_time_2_1[i]=alaram_time_3_1[i];
      }
     for(i=0;i<2;i++)
      {
      alaram_time_2_2[i]=alaram_time_3_2[i];
      }
     for(i=0;i<2;i++)
      {
      alaram_time_2_3[i]=alaram_time_3_3[i];
      }
     for(i=0;i<2;i++)
      {
      alaram_time_2_4[i]=alaram_time_3_4[i];
      }
     alaram_time2_h=alaram_time3_h;
     alaram_time2_m=alaram_time3_m;
     alaram_time2_d=alaram_time3_d;
     alaram_time2_o=alaram_time3_o;
     alaram_count--;
     alaram3sec=0;
     //alaram1();
     alaram2();
     alaram3_lcd_flag = 0;
    }


     if(alaram_count == 4)
    {
     desc_count2=desc_count3;
```

```
for(i=0;i<desc_count3;i++)
{
alaram_des_2[i]=alaram_des_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_1[i]=alaram_time_3_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_2[i]=alaram_time_3_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_3[i]=alaram_time_3_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_2_4[i]=alaram_time_3_4[i];
}
alaram_time2_h=alaram_time3_h;
alaram_time2_m=alaram_time3_m;
alaram_time2_d=alaram_time3_d;
alaram_time2_o=alaram_time3_o;


desc_count3=desc_count4;
for(i=0;i<desc_count4;i++)
{
alaram_des_3[i]=alaram_des_4[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_1[i]=alaram_time_4_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_2[i]=alaram_time_4_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_3[i]=alaram_time_4_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_4[i]=alaram_time_4_4[i];
}
alaram_time3_h=alaram_time4_h;
alaram_time3_m=alaram_time4_m;
alaram_time3_d=alaram_time4_d;
```

```
 alaram_time3_o=alaram_time4_o;
 alaram_count--;
 alaram4sec=0;
// alaram1();
 alaram2();
 alaram3();
 alaram4_lcd_flag = 0;
}

 if(alaram_count == 5)
{

 desc_count2=desc_count3;
 for(i=0;i<desc_count3;i++)
 {
 alaram_des_2[i]=alaram_des_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_1[i]=alaram_time_3_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_2[i]=alaram_time_3_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_3[i]=alaram_time_3_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_2_4[i]=alaram_time_3_4[i];
 }
 alaram_time2_h=alaram_time3_h;
 alaram_time2_m=alaram_time3_m;
 alaram_time2_d=alaram_time3_d;
 alaram_time2_o=alaram_time3_o;


 desc_count3=desc_count4;
 for(i=0;i<desc_count4;i++)
 {
 alaram_des_3[i]=alaram_des_4[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_1[i]=alaram_time_4_1[i];
 }
 for(i=0;i<2;i++)
 {
```

```
 alaram_time_3_2[i]=alaram_time_4_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_3[i]=alaram_time_4_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_3_4[i]=alaram_time_4_4[i];
}
alaram_time3_h=alaram_time4_h;
alaram_time3_m=alaram_time4_m;
alaram_time3_d=alaram_time4_d;
alaram_time3_o=alaram_time4_o;


desc_count4=desc_count5;
for(i=0;i<desc_count5;i++)
{
alaram_des_4[i]=alaram_des_5[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_1[i]=alaram_time_5_1[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_2[i]=alaram_time_5_2[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_3[i]=alaram_time_5_3[i];
}
for(i=0;i<2;i++)
{
 alaram_time_4_4[i]=alaram_time_5_4[i];
}
alaram_time4_h=alaram_time5_h;
alaram_time4_m=alaram_time5_m;
alaram_time4_d=alaram_time5_d;
alaram_time4_o=alaram_time5_o;



alaram_count--;
alaram5sec=0;
//alaram1();
alaram2();
alaram3();
alaram4();
```

```
    alaram5_lcd_flag = 0;
   }
}


 if(alaram3_occured_flag==1)
{

    if(alaram_count == 3)
   {
    alaram_count--;
    alaram3sec=0;
    //alaram1();
    //alaram2();
    alaram3_lcd_flag = 0;
   }


    if(alaram_count == 4)
   {

    desc_count3=desc_count4;
    for(i=0;i<desc_count4;i++)
    {
    alaram_des_3[i]=alaram_des_4[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_3_1[i]=alaram_time_4_1[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_3_2[i]=alaram_time_4_2[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_3_3[i]=alaram_time_4_3[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_3_4[i]=alaram_time_4_4[i];
    }
    alaram_time3_h=alaram_time4_h;
    alaram_time3_m=alaram_time4_m;
    alaram_time3_d=alaram_time4_d;
    alaram_time3_o=alaram_time4_o;
    alaram_count--;
    alaram4sec=0;
   // alaram1();
   // alaram2();
 alaram3();
 alaram4_lcd_flag = 0;
}

 if(alaram_count == 5)
{




 desc_count3=desc_count4;
 for(i=0;i<desc_count4;i++)
 {
 alaram_des_3[i]=alaram_des_4[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_1[i]=alaram_time_4_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_2[i]=alaram_time_4_2[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_3[i]=alaram_time_4_3[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_3_4[i]=alaram_time_4_4[i];
 }
 alaram_time3_h=alaram_time4_h;
 alaram_time3_m=alaram_time4_m;
 alaram_time3_d=alaram_time4_d;
 alaram_time3_o=alaram_time4_o;


 desc_count4=desc_count5;
 for(i=0;i<desc_count5;i++)
 {
 alaram_des_4[i]=alaram_des_5[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_4_1[i]=alaram_time_5_1[i];
 }
 for(i=0;i<2;i++)
 {
  alaram_time_4_2[i]=alaram_time_5_2[i];
 }
```

```
    for(i=0;i<2;i++)
    {
     alaram_time_4_3[i]=alaram_time_5_3[i];
    }
    for(i=0;i<2;i++)
    {
     alaram_time_4_4[i]=alaram_time_5_4[i];
    }
    alaram_time4_h=alaram_time5_h;
    alaram_time4_m=alaram_time5_m;
    alaram_time4_d=alaram_time5_d;
    alaram_time4_o=alaram_time5_o;



    alaram_count--;
    alaram5sec=0;
    //alaram1();
    //alaram2();
    alaram3();
    alaram4();
    alaram5_lcd_flag = 0;
   }
}




      if(alaram4_occured_flag==1)
{

    if(alaram_count == 4)
   {


    alaram_count--;
    alaram4sec=0;
   // alaram1();
   // alaram2();
   // alaram3();
    alaram4_lcd_flag = 0;
   }

    if(alaram_count == 5)
   {


    desc_count4=desc_count5;
    for(i=0;i<desc_count5;i++)
    {
            alaram_des_4[i]=alaram_des_5[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_4_1[i]=alaram_time_5_1[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_4_2[i]=alaram_time_5_2[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_4_3[i]=alaram_time_5_3[i];
            }
            for(i=0;i<2;i++)
            {
             alaram_time_4_4[i]=alaram_time_5_4[i];
            }
            alaram_time4_h=alaram_time5_h;
            alaram_time4_m=alaram_time5_m;
            alaram_time4_d=alaram_time5_d;
            alaram_time4_o=alaram_time5_o;



            alaram_count--;
            alaram5sec=0;
            alaram4();
            alaram5_lcd_flag = 0;
           }
       }



       if(alaram5_occured_flag==1)
       {
            if(alaram_count == 5)
           {
            alaram_count--;
            alaram5sec=0;
            alaram5_lcd_flag = 0;
           }
       }

     }
   //Printf("\r\n i am going out\n\r");
}
```

```
/*------------------------------------------------------------------------------------------------------------
----------------------------------------------------
 * Raghunath Reddy
 * ECEN 5613, Keypad (4*3) using scanning algorithm
 * Fall 2016,Prof. Mc Clure
 * University of Colorado at Boulder
 * --------------------------------------
 * This file helps the user to use keypad not only to enter numbers but also alphabets by
changing modes using *,# and also touchscreens certain functions are
 included in this file.
 * ------------------------------------------------------------------------------------------------------------
---------------------------------------------------- */


/* Gloabalized variables---------------------------------------------------- */

 unsigned char changemode=0;                     //Indicates the mode of the keypad.
 unsigned char keypress_count=0;                  //Counts the number of keypressed
 unsigned char page2_flag=0;                    //Page2_flag  tells you if the code is in page 2
of the main code
 unsigned char touchpressed_done_flag=0;             //Tells us if the done button is pressed on
the touchscreen using touchscreen.h

 /* Prototypes ---------------------------------------------------- */
 /*------------------------------------------------------------------------------------------------------------
----------------------------------------------------
  * keypad_init()
  * Purpose:     Initializing the keypad with 3pins as output and 4 pins as inputs
  * Algorithm:   Initialize pin5,pin6,pin7 as outputs and pin8,pin9,pin10,pin11 as inputs for
performing a scanning function in the later part of code
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */
 void keypad_init();


 void keypress();
 /*------------------------------------------------------------------------------------------------------------
----------------------------------------------------
  * keypress()
  * Purpose:     Waits for the key to pressed and than updates using scanning algorithm
  * Algorithm:
  * Calcuations: None
  * Return:      None
  *------------------------------------------------------------- */

 void adc_read();

void keypad_init()
{
  LPC_PINCON -> PINSEL3 &= ~((1<<5)|(1<<4));                                  //Pin to
LED1 by mbed development board initalizes it as gpio
  LPC_GPIO1-> FIODIR |=     (1<<18);                                      //Make LED1 as
output
  LPC_GPIO1-> FIODIR |= (1<<20);                                          //output

  LPC_PINCON -> PINSEL0&= ~((1<<19)|(1<<18));                                  //Pin5 as
gpio
  LPC_PINCON -> PINSEL0&= ~((1<<17)|(1<<16));                                  //Pin6 as
gpio
  LPC_PINCON -> PINSEL0&= ~((1<<15)|(1<<14));                                  //Pin7 as
gpio


  LPC_PINCON -> PINSEL0 &= ~((1<<13)|(1<<12));                                 //Pin8 as
gpio
  LPC_PINCON -> PINSEL0 &= ~((1<<1)|(1<<0));                                  //Pin9 as
gpio
  LPC_PINCON -> PINSEL0 &= ~((1<<2)|(1<<3));                                  //Pin10 as
gpio
  LPC_PINCON -> PINSEL1 &= ~((1<<5)|(1<<4));                                  //Pin11 as
gpio


  LPC_GPIO0-> FIODIR |= (1<<9);                                          //Pin5 as output
  LPC_GPIO0-> FIODIR |= (1<<8);                                          //Pin6 as output
  LPC_GPIO0-> FIODIR |= (1<<7);                                          //Pin7 as input


  LPC_GPIO0-> FIODIR &= ~(1<<6);                                           //Pin8 as input
  LPC_GPIO0-> FIODIR &= ~(1<<0);                                           //Pin9 as input
  LPC_GPIO0-> FIODIR &= ~(1<<1);                                           //Pin10 as input
  LPC_GPIO0-> FIODIR &= ~(1<<18);                                           //Pin11 as
input

  LPC_GPIO0 -> FIOCLR |= (1<<9);                                          //Pin5 is made
low initially
  LPC_GPIO0 -> FIOCLR |= (1<<8);                                          //Pin6 is made
low initally
  LPC_GPIO0 -> FIOCLR |= (1<<7);                                          //Pin7 is made
low initally


  LPC_GPIO0-> FIODIR |= (1<<17);                                          //Mode0 LED as
an output
  LPC_GPIO0-> FIODIR |= (1<<15);                                          //Mode1 LED as
an output
  LPC_GPIO0-> FIODIR |= (1<<16);                                          //Mode2 LED as
an output
```

```
   LPC_GPIO2-> FIODIR |= (1<<3);                                          //Mode3 LED as
an output


   LPC_GPIO2-> FIODIR |= (1<<4);//output
   LPC_GPIO2 -> FIOCLR |= (1<<4);


   //mode 0
   LPC_GPIO0 -> FIOSET |= (1<<17);                                         //Making Mode
0 LED high
   LPC_GPIO0 -> FIOCLR |= (1<<15);
   LPC_GPIO0 -> FIOCLR |= (1<<16);
   LPC_GPIO2 -> FIOCLR |= (1<<3);


}

/*
void key_press
Algorithm:
1.Initally make all the input pins low
2.check if there is an input in any of them if yes than implement the software debounce
which is checking input after a certain delay, if no than bo back and wait for the input.
3.Now check if the value1 is still present.If yes
i.   Make colum1 zero and again check if the value at row1 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
ii.  Make colum2 zero and again check if the value at row1 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
ii.  Make colum3 zero and again check if the value at row1 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.

4.Now check if the value2 is still present.If yes
i.   Make colum1 zero and again check if the value at row2 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
ii.  Make colum2 zero and again check if the value at row2 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
ii.  Make colum3 zero and again check if the value at row2 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.

5.Now check if the value3 is still present.If yes
i.   Make colum1 zero and again check if the value at row3 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
```

```
ii.  Make colum2 zero and again check if the value at row3 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
ii.  Make colum3 zero and again check if the value at row3 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.

6.Now check if the value4 is still present.If yes
i.   Make colum1 zero and again check if the value at row4 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
ii.  Make colum2 zero and again check if the value at row4 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
ii.  Make colum3 zero and again check if the value at row4 if the value still exists we can
update the value in the global variable and change the keypress count with an
inital inspection of which mode it is in and the next step is point7.
7.Led indication of exiting loop
*/

void key_press()

{
   unsigned char Value1;                                          //Value1 helps in telling if
there is any input at row1
   unsigned char Value2;                                          //Value2 helps in telling if
there is any input at row2
   unsigned char Value3;                                          //Value3 helps in telling if
there is any input at row3
   unsigned char Value4;                                          //Value4 helps in telling if
there is any input at row4
   unsigned char keypress=0;                                        //Flag to check if the
keypressed is pressed
   unsigned char i;

   while(keypress==0)
   {
     //Make column1,column2,column3 as inputs
     LPC_GPIO0 -> FIOCLR |= (1<<9);
     LPC_GPIO0 -> FIOCLR |= (1<<8);
     LPC_GPIO0 -> FIOCLR |= (1<<7);

     //Read the four columns
     Value1 = ((LPC_GPIO0 -> FIOPIN & (1<<6)) >> 6);
     Value2 = ((LPC_GPIO0 -> FIOPIN & (1<<0)) >> 0);
     Value3 = ((LPC_GPIO0 -> FIOPIN & (1<<1)) >> 1);
     Value4 = ((LPC_GPIO0 -> FIOPIN & (1<<18)) >> 18);

     //check if any key is pressed
     if((Value1 && Value2 && Value3 && Value4) == 0 )
```

```
      {
        //give a delay of certain time and read the values of the rows
        wait(0.1);
        Value1 = ((LPC_GPIO0 -> FIOPIN & (1<<6)) >> 6);
        Value2 = ((LPC_GPIO0 -> FIOPIN & (1<<0)) >> 0);
        Value3 = ((LPC_GPIO0 -> FIOPIN & (1<<1)) >> 1);
        Value4 = ((LPC_GPIO0 -> FIOPIN & (1<<18)) >> 18);
        //Check if the key is still pressed
        if ((Value1 && Value2 && Value3 && Value4) == 0 )
          {
         if(Value1 == 0)                                   //check if any key in row1 is
pressed.
           { //Make only column3 low
             LPC_GPIO0 -> FIOSET |= (1<<7);
             LPC_GPIO0 -> FIOSET |= (1<<8);
             LPC_GPIO0 -> FIOCLR |= (1<<9);                          //column 1 low


             Value1 = ((LPC_GPIO0 -> FIOPIN & (1<<6)) >> 6);                //read row1
value again
             if (Value1 == 0)                              //check if key1  is pressed
             {
               if (changemode == 1)                           //check the mode of the
keypad
               {
                 if(column==0)                             //backspace function
                 {
                    // if back space is pressed when it is at the end of the page during the cornrer
conditions
                    column = 126;
                    if(pagevalue == 0)
                    {
                       pagevalue = 7;
                       lcd_cmd(0xB0| page[pagevalue]);
                    }
                    else
                    {
                       pagevalue = pagevalue - 1;
                       lcd_cmd(0xB0| page[pagevalue]);
                    }

                 }


                 column=column-6;                               //changes te position of the
cursor
                 lcd_cursor_column_flag=1;                          //Flag to indicate that we
dont require any cursor blink
                 lcd_changelocation();
                 lcd_cursor_column_flag=0;
```

```
            lcd_write_character(0x00);                        //0x00 is written into the
LCD has cursor blink can be erased

          }
          else if(changemode == 0)
          {
          lcd_write_character('1');                           //if mode 0 than print 1 on to
the LCD
          keypress=1;
          }
          else if(changemode ==2)                             //Mode 2 : enter next line
          {
           if(pagevalue == 7)
           {
              pagevalue=0;                                   //corner conditions
           }
           else
           {
              pagevalue=pagevalue+1;
           }
           lcd_cmd(0xB0|page[pagevalue]);
           column=0x00;
           column_change();
          }
          else if(changemode ==3)                             //Mode3: write 1 into the
LCD
          {
            lcd_write_character('1');
            keypress=1;
          }
        }
        else
        {
         //Check the column2 keys are pressed
         LPC_GPIO0 -> FIOCLR |= (1<<8);
         LPC_GPIO0 -> FIOSET |= (1<<7);
         LPC_GPIO0 -> FIOSET |= (1<<9);
         Value1 = ((LPC_GPIO0 -> FIOPIN & (1<<6)) >> 6);                  //read value
of colum2
         if (Value1 == 0)
           {
             keypress=1;
             if(changemode==0)                               // to enter charcter 2
             {
              lcd_write_character('2');
             }
             else if (changemode==1)                         // to enter charcter A
             {
               lcd_write_character('A');
             }
```

```
              else if (changemode==2)                              // to enter charcter B
              {
                lcd_write_character('B');
              }
              else if (changemode==3)                               // to enter charcter C
              {
                lcd_write_character('C');
              }
            }
          else
            {
            //clear column3
              LPC_GPIO0 -> FIOCLR |= (1<<7);
              LPC_GPIO0 -> FIOSET |= (1<<8);
              LPC_GPIO0 -> FIOSET |= (1<<9);
              Value1 = ((LPC_GPIO0 -> FIOPIN & (1<<6)) >> 6); //read the value od row1
               {
                keypress=1;
                if(changemode==0)
              {
               lcd_write_character('3');                            // to enter charcter 3
              }
              else if (changemode==1)
              {
                lcd_write_character('D');                             // to enter charcter D
              }
              else if (changemode==2)
              {
                lcd_write_character('E');                             // to enter charcter E
              }
              else if (changemode==3)
              {
                lcd_write_character('F');                             // to enter charcter F
              }

             }
            }
          }
        }
        //This similar to value1 equal to zero ,i.e check if row2 pressed if yes what make the
column1,column2,column3 zero in the same order as it is
        //done for value1=0
        else if(Value2 == 0)
        {
         LPC_GPIO0 -> FIOSET |= (1<<7);
         LPC_GPIO0 -> FIOSET |= (1<<8);
          LPC_GPIO0 -> FIOCLR |= (1<<9);

         Value2 = ((LPC_GPIO0 -> FIOPIN & (1<<0)) >> 0);
         if (Value2 == 0)
```

```
            {
             keypress=1;

             if(changemode==0)
                 {
                  lcd_write_character('4');                              // to enter charcter
4
                 }
                 else if (changemode==1)
                 {
                   lcd_write_character('G');                             // to enter charcter
G
                 }
                 else if (changemode==2)
                 {
                   lcd_write_character('H');                             // to enter charcter
H
                 }
                 else if (changemode==3)
                 {
                   lcd_write_character('I');                             // to enter charcter I
                 }

            }
            else
            {

             LPC_GPIO0 -> FIOCLR |= (1<<8);
             LPC_GPIO0 -> FIOSET |= (1<<7);
             LPC_GPIO0 -> FIOSET |= (1<<9);
             Value2 = ((LPC_GPIO0 -> FIOPIN & (1<<0)) >> 0);
             if (Value2 == 0)
               {
                 keypress=1;
                 if(changemode==0)
                 {
                  lcd_write_character('5');                 // to enter charcter 5 on LCD
                 }
                 else if (changemode==1)
                 {
                   lcd_write_character('J');                // to enter charcter J on LCD
                 }
                 else if (changemode==2)
                 {
                   lcd_write_character('K');                // to enter charcter K on LCD
                 }
                 else if (changemode==3)
                 {
                   lcd_write_character('L');                // to enter charcter L on LCD
                 }
```

```
                  }
               else
                  {
                     LPC_GPIO0 -> FIOCLR |= (1<<7);
                     LPC_GPIO0 -> FIOSET |= (1<<8);
                     LPC_GPIO0 -> FIOSET |= (1<<9);
                     Value2 = ((LPC_GPIO0 -> FIOPIN & (1<<0)) >> 0);
                     if (Value2 == 0)
                       {
                         keypress=1;
                         if(changemode==0)
                     {
                       lcd_write_character('6');                              // to enter charcter 6 on LCD
                     }
                     else if (changemode==1)
                     {
                        lcd_write_character('M');                             // to enter charcter M on LCD
                     }
                     else if (changemode==2)
                     {
                        lcd_write_character('N');                             // to enter charcter N on LCD
                     }
                     else if (changemode==3)
                     {
                        lcd_write_character('O');                              // to enter charcter 0 on LCD
                     }
                        }
                  }
               }
             }
             //This similar to value1 equal to zero ,i.e check if row2 pressed if yes what make the
column1,column2,column3 zero in the same order as it is
             //done for value1=0
            else if(Value3 == 0)
            {
              LPC_GPIO0 -> FIOSET |= (1<<7);
              LPC_GPIO0 -> FIOSET |= (1<<8);
              LPC_GPIO0 -> FIOCLR |= (1<<9);
              Value3 = ((LPC_GPIO0 -> FIOPIN & (1<<1)) >> 1);
              if (Value3 == 0)
              {
                keypress=1;
                if(changemode==0)
                {
                  lcd_write_character('7');                                              // to enter charcter 0 on
LCD
                }
                else if (changemode==1)
                {
```

```
            lcd_write_character('P');                               // to enter charcter P on
LCD
           }
           else if (changemode==2)
           {
                lcd_write_character('R');                           // to enter charcter R on
LCD
           }
              else if (changemode==3)
              {
                lcd_write_character('S');                           // to enter charcter S on
LCD
              }
          }
          else
          {

          LPC_GPIO0 -> FIOCLR |= (1<<8);
          LPC_GPIO0 -> FIOSET |= (1<<7);
          LPC_GPIO0 -> FIOSET |= (1<<9);
          Value3 = ((LPC_GPIO0 -> FIOPIN & (1<<1)) >> 1);
          if (Value3 == 0)
            {
              keypress=1;
              if(changemode==0)
              {
               lcd_write_character('8');                            // to enter charcter 8 on
LCD
              }
              else if (changemode==1)
              {
                lcd_write_character('T');                           // to enter charcter T on
LCD
              }
              else if (changemode==2)
              {
                lcd_write_character('U');                           // to enter charcter U on
LCD
              }
              else if (changemode==3)
              {
                lcd_write_character('V');                          // to enter charcter V on
LCD
              }
              //LPC_GPIO1 -> FIOSET |= (1<<18);
              //wait(1);
              //LPC_GPIO1 -> FIOCLR |= (1<<18);
            }
          else
            {
                 LPC_GPIO0 -> FIOCLR |= (1<<7);
                 LPC_GPIO0 -> FIOSET |= (1<<8);
                 LPC_GPIO0 -> FIOSET |= (1<<9);
                 Value3 = ((LPC_GPIO0 -> FIOPIN & (1<<1)) >> 1);
                 if (Value3 == 0)
                   {
                    keypress=1;
                    if(changemode==0)
                 {
                   lcd_write_character('9');                              // to enter charcter 9 on
LCD
                 }
                 else if (changemode==1)
                 {
                   lcd_write_character('W');                              // to enter charcter W on
LCD
                 }
                 else if (changemode==2)
                 {
                   lcd_write_character('X');                              // to enter charcter X on
LCD
                 }
                 else if (changemode==3)
                 {
                   lcd_write_character('Y');                              // to enter charcter Y on
LCD
                 }

                   }
               }
            }
          }

          else if(Value4 == 0)
          {
            LPC_GPIO0 -> FIOSET |= (1<<7);
            LPC_GPIO0 -> FIOSET |= (1<<8);
             LPC_GPIO0 -> FIOCLR |= (1<<9);

            Value4 = ((LPC_GPIO0 -> FIOPIN & (1<<18)) >> 18);
            if (Value4 == 0)
            {
              if(changemode ==0)                                          //Mode change buttons
              {
               changemode=1;
                  //mode 1    LEDs change color based on Mode
                LPC_GPIO0 -> FIOCLR |= (1<<17);
                LPC_GPIO0 -> FIOSET |= (1<<15);
                LPC_GPIO0 -> FIOCLR |= (1<<16);
                LPC_GPIO2 -> FIOCLR |= (1<<3);
            }
            else
            {
            changemode=0;
               //mode 0
             LPC_GPIO0 -> FIOSET |= (1<<17);
             LPC_GPIO0 -> FIOCLR |= (1<<15);
             LPC_GPIO0 -> FIOCLR |= (1<<16);
             LPC_GPIO2 -> FIOCLR |= (1<<3);
            }

          }
          else
          {

           LPC_GPIO0 -> FIOCLR |= (1<<8);
           LPC_GPIO0 -> FIOSET |= (1<<7);
           LPC_GPIO0 -> FIOSET |= (1<<9);
           Value4 = ((LPC_GPIO0 -> FIOPIN & (1<<18)) >> 18);

             if (Value4 == 0)
              {


                 if(changemode==0)
                 {
                   lcd_write_character('0');
                   keypress=1;
                 }
                 else if (changemode==1)
                 {
                    lcd_write_character('Q');                              // to enter charcter Q on LCD
                    keypress=1;
                 }
                 else if (changemode==2)
                 {
                    lcd_write_character('Z');                              // to enter charcter Z on LCD
                    keypress=1;
                 }
                 else if (changemode==3)
                 {
                    lcd_write_character(0x00);                               // to enter charcter space
on LCD
                    keypress=1;
                 }
              }
           else
              {
                 LPC_GPIO0 -> FIOCLR |= (1<<7);
                 LPC_GPIO0 -> FIOSET |= (1<<8);
                 LPC_GPIO0 -> FIOSET |= (1<<9);
                 Value4 = ((LPC_GPIO0 -> FIOPIN & (1<<18)) >> 18);

              if (Value4 == 0)                                      //changing the mode from
mode2, mode3
                 {
                      LPC_GPIO1 -> FIOCLR |= (1<<18);
                      wait(0.1);
                     LPC_GPIO1 -> FIOSET |= (1<<18);
                     wait(0.1);

                   if(changemode == 2)
                     {
                     changemode=3;
                        //mode 3
                      LPC_GPIO0 -> FIOCLR |= (1<<17);
                      LPC_GPIO0 -> FIOCLR |= (1<<15);
                      LPC_GPIO0 -> FIOCLR |= (1<<16);
                      LPC_GPIO2 -> FIOSET |= (1<<3);

                     }
                   else
                     {
                     changemode=2;
                        //mode 2
                      LPC_GPIO0 -> FIOCLR |= (1<<17);
                      LPC_GPIO0 -> FIOCLR |= (1<<15);
                      LPC_GPIO0 -> FIOSET |= (1<<16);
                      LPC_GPIO2 -> FIOCLR |= (1<<3);
                     }

                   }
              }
           }
         }
        }
      }



     // Check if the done button the touch screen is pressed
     xcor=0;
      ycor=0;

      if(page2_flag==1)       //only when keypress dunction is used in page2
   {
     for(i=0;i<5;i++)
     {
          adc_read();
          if(((xcor > 0x5200 && xcor< 0x7400)&&(ycor >0x7100 && ycor< 0x7C00)))
//Previously determined x nd y coordinates for done button on touchscreen
          {
           touchpressed_done_flag=1;
           goto endloop1;
          }
        }
        }

}


        endloop1:

        //LED indication for verifiacation
        LPC_GPIO1 -> FIOCLR |= (1<<18);
        wait(0.1);
        LPC_GPIO1 -> FIOSET |= (1<<18);
        wait(0.1);
        LPC_GPIO1 -> FIOCLR |= (1<<18);
        wait(0.1);


}



/*--------------------------------------------------------------------------------------------------------
------------------------------------------------------
 * Raghunath Reddy
 * ECEN 5613, 4 wire resistive touch screen
 * Fall 2016,Prof. Mc Clure
 * University of Colorado at Boulder
 * -------------------------------------
 * This file lets the user interface a four wire resistive touch screen by using the ADC.
 * --------------------------------------------------------------------------------------------------------
---------------------------------------------------- */
int xcor;
int ycor;
char timevalue[16]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

int readX()
{
  AnalogIn yaxis2(A3);
  DigitalOut xaxis1(P0_23); //Port 0, pin 10
  AnalogIn yaxis3(A1);
  DigitalOut xaxis2(P0_25); //Port 0, pin 10
  xaxis2=0;//make x2 low
  xaxis1=1;//make x1 high
  wait_ms(5); //pause to allow lines to power up
  int adcy2= yaxis2.read_u16();
  return adcy2;
}

int readY(){
  int adcx2;
  AnalogIn x1(A0);
  DigitalOut y2(P0_26); //Port 0, pin 10
  AnalogIn x2(A2);
  DigitalOut y3(P0_24); //Port 0, pin 10
  y2=0;//make y1 low
  y3=1;//make y2 high
  wait_ms(5); //pause to allow lines to power up
  adcx2= x2.read_u16();
  return adcx2;
}

void adc_read()
{
  int xtemp[5];
  int ytemp[5];
  int x=0;
  int y=0;
  unsigned char i=0;
  int temp_check;
  temp_check =readX();
  while(i<5 && temp_check > 200)
  {
  xtemp[i] = readX();
  ytemp[i] = readY();
  i++;
  if(i==5)
  {
  for(i=0;i<5;i++)
  {
      x=x+xtemp[i];
      y=y+ytemp[i];
  }
  x=x/5;
  y=y/5;
  xcor=x;
  ycor=y;
  }

  }
  i=0;
  x=0;
  y=0;
}
```

8.4 Appendix-Datasheets

1. LPC1768 User manual — http://www.nxp.com/documents/user_manual/UM10360.pdf

2. Graphic LCD — https://www.adafruit.com/datasheets/TG12864H3-05A%20EN_V1.0.pdf

3. Resistive touchscreen — https://cdn.sparkfun.com/datasheets/LCD/Accessories/HOW%20DOES%20IT%20WORK.pdf