VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Samuel Farris Mamo

HOME AUTOMATION USING TOUCH SCREEN

Technology and Communication
2014

ACKNOWLEDGEMENT

Firstly, I would like to express my great appreciation to my thesis supervisor, Jukka Matila, for his instruction to my thesis as well as my academic research direction. During my thesis working period, he can always guide me to solve the problem in his distinctive way.

In addition, I would like to thank to my family for their support and encouragement during my studies.

Vaasa, Finland

5.6.2014

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Samuel Mamo
Title	Home Automation Using Touch Screen
Year	2014
Language	English
Pages	37 + 19 Appendices
Name of Supervisor	Jukka Matila

The general objective of this thesis was to make a multipurpose standalone device capable of controlling peripherals, such as curtain, lighting, central A/C system, socket, door, CCTV camera and home appliance, etc.

The objective of this thesis was to monitor room temperature using user friendly Touch User Interface and an Ethernet web browser. This project was based on LPC1788 Development kit (7 inch Touch Screen). TUI was uses emWin library, RS485 serial port and Vbus protocol is used for communicated between Development kit and for SorTech AG cooling system.

The development environment was LPCXpresso, Bitmap Converter for emWin Demo Version and Font Converter for emWin Demo Version those are Microsoft Windows platform. The most useful in this project was LPCXpresso.

This thesis accomplished two tasks. The Touch Interface with its full functionality has had its design and coding completed. The communication protocol that transmits commands between the development kit and the SorTech Cooling System is partially completed. Subsequently, the development kit is connected to a desktop computer which is uses to launch Putty software as a communication application to execute the sample data sent to it.

Keyword Touch Screen, emWin library, RS485, Vbus, C, SorTech.

CONTENTS

LIST OF FIGURES AND TABLES

1 INTRODUCTION

1.1 General Background

A home automation system is a term widely used to describe household amenities and appliances that are programmed to work together and it is to provide improved convenience, comfort, energy efficiency and security; it is increased quality of life for persons /1/ and decrease power consumption. Home Automation can be centrally controlled by an LCD panel (touch screen) or Ethernet web base interface.

Figure 1. LCD panel Control panel, able to control thermostat, lighting and home entertainment. /1/

By far, touch screens are the most common, innovative and user friendly to use. They can be deployed to automate home lights, security panels, HVAC (heating, ventilation and air-conditioning) /1/ or they may work as a standalone controller that controls a range of automated devices including door gates and SorTech AG cooling systems.

A touch screen is basically a miniature computer screen that executes commands following fingers touching keys on the screen; in the same manner as clicking with a mouse on a desktop computer. The touch screen has visual buttons, usually, that either causes an event to be effected (such as a light going on or off, or the thermostat put on or off) or other menus displayed as needed in the automation. It is also not uncommon for a touch screen to display video, thereby allowing the user to monitor mounted on doors, in rooms, and in premises surrounding a property.

1.2 Purpose of Thesis

The main objective of a project was to design the control of the room temperature on Touch User Interface (**TUI**) so that the user is able to adjust desired temperature by using Touch User Interface and the measured temperature is constantly informed on the device screen or on Ethernet. LPC1788 DVK is also informed current temperature at its mounting place (room).The project was implemented at Technobotnia.This project consists of the following hardware and software:

- SorTech AG cooling system
- LPC1788 DVK
- LCD panel
- emWin library
- Touch User Interface
- Serial port commutation

Figure 2. shows the detailed structure of a project hardware and TUI.

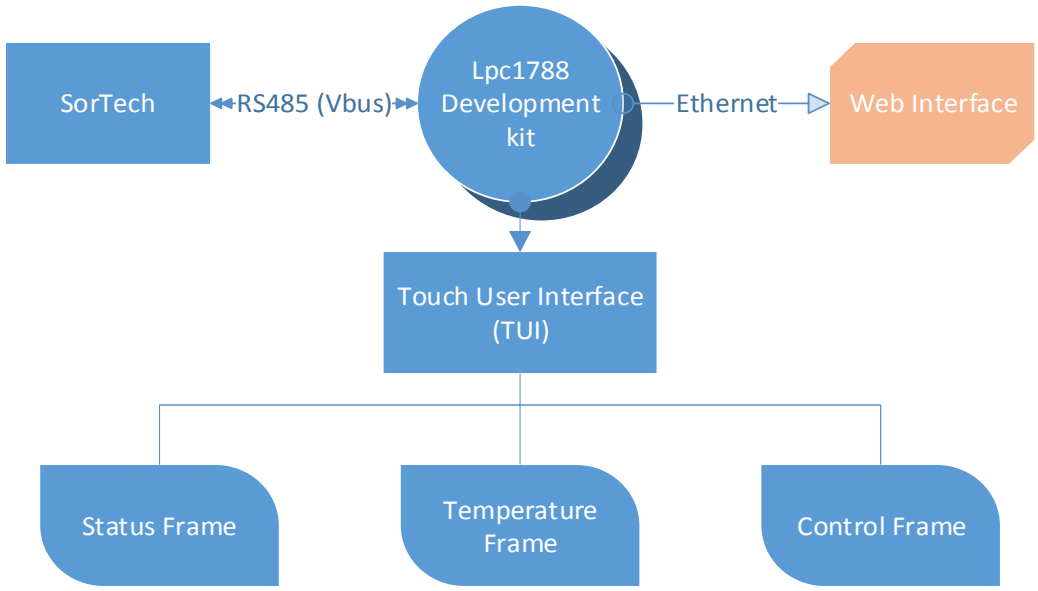

Figure 2. Block diagram of a project.

The SorTech AG cooling system, LPC1788 DVK, Touch panel, TUI, emWin lib and Serial port commutation will be discussed in the following chapter sections.

1.3 Overview Structure of Thesis

This thesis is divided into 9 chapters. The first chapter introduces the background information of this thesis. The second chapter presents background of SorTech cooling system, including their functionality and component. The third chapter presents the LPC1788 DVK background and feature. The fourth chapter presents the emWin background and functionality. The fifth chapter explains the structure of TUI. The sixth chapter presents Communication and VBus protocol. The seventh chapter presents implementation. The eighth chapter presents Testing. The last chapter presents conclusion.

2 SORTESCH

The Adsorption Chiller Silica gel is a design by SorTech; ACS uses thermal driving energy for cooling at a low power range. In places where there is solar heating, district heat, process heating or any kind of waste heating, the devise works and is a good alternative. It is more environmentally friendly and cheaper to operate than conventional compression chillers. /2/

2.1 The Complete Thermal Cooling System

"The complete system for **ACS** operation is comprised of three hydraulic circuits:

- The driving circuit, HT, which delivers heat at a high level for desorption (e.g. solar heat, heat from a combined heat and power plant, district heat, industrial waste heat)
- The re-cooling circuit, MT, which removes heat at a medium or surrounding temperature level (e.g. dry re-cooler with sprinkler option (RCS), swimming pool heater, geothermal probe, wet cooling tower, lake heat exchanger
- The cold water circuit, LT, for distribution and usage of the generated cold (e.g. cooling ceilings, wall / floor cooling, fan coils, concrete core activation, etc.)) "/2/

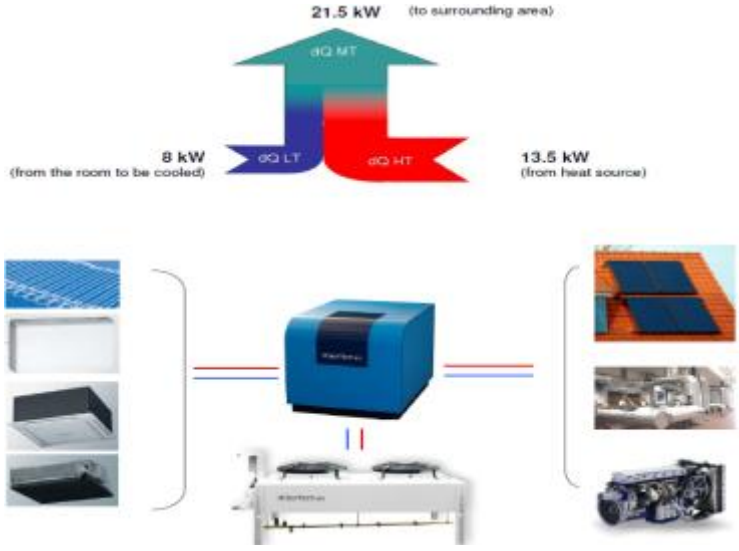

Figure 3. SorTech and SorTech Appliance /2/

2.2 ACS System Interfaces

The wiring of the ACS by the installer includes

- The mains supply
- Control Signals
- Temperature Sensors
- Device feedback

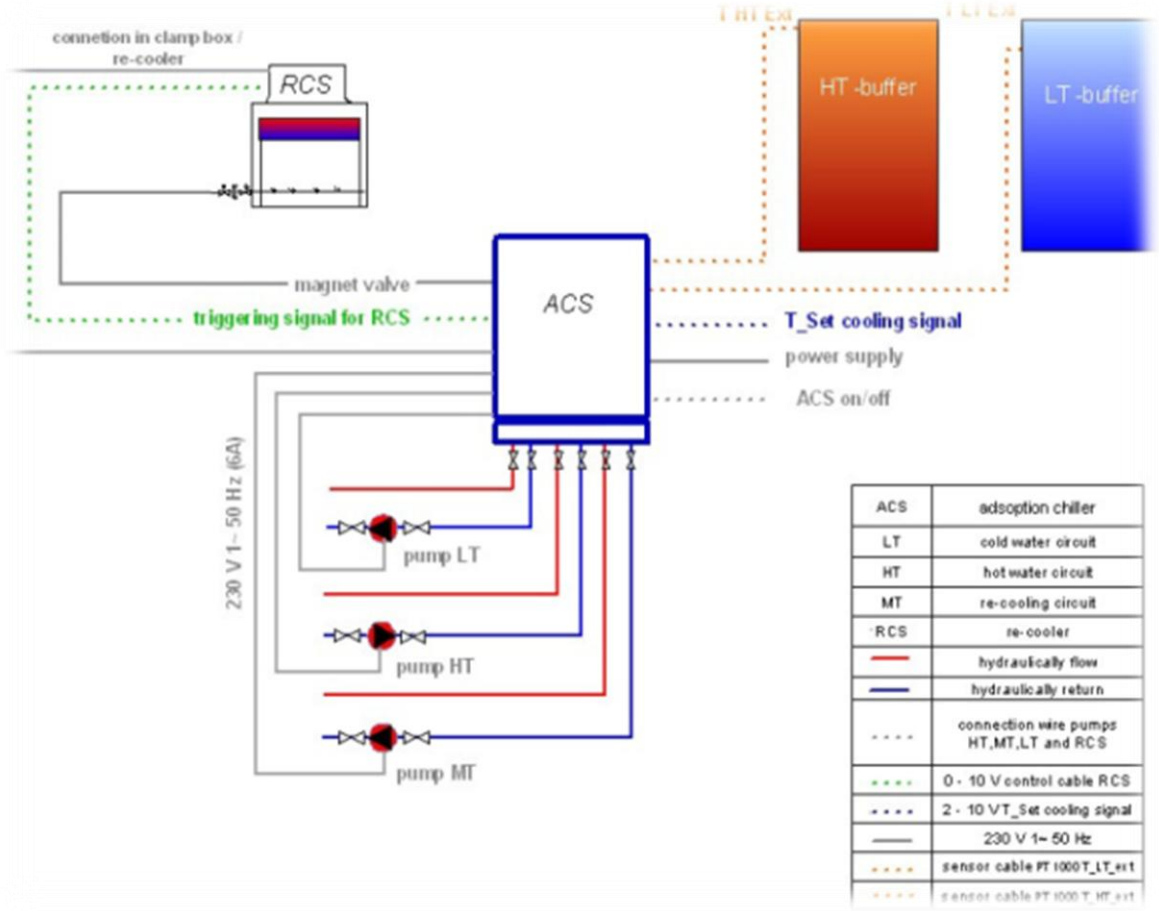

Figure 4. Wiring of the ACS /2/

3 DEVELOPMENT KIT

The complete kit, LPC1788 Development Kit, comes with all needed assortment of hardware and software that you can use to design, integrate and test any applications of your choice. Embedded Artists are provided this development kit and LCD Panel.

3.1 LPC1788

The LPC1788 Evaluation Board is built-in with Cortex-M3 MCU which can be used for a multitude of industrial application that require advanced communication and high quality graphic displays in addition to high level of integration and low power consumption at frequencies of 120 MHz. /3/ “It can support optimized graphic LCD display controller interfaces directly with a variety of color and monochrome LCD panels with resolutions up to 1024x768 pixels, and includes free emWin graphic libraries.” /4/ In this project the following features will be used:

- LCD panel
- emWin graphic libraries: uses of GUI.
- UART: it allow the communication between the MUC and the RS485
- I2C: is connected to LM75 temperatures sensor
- RS485 interface
- Ethernet interface

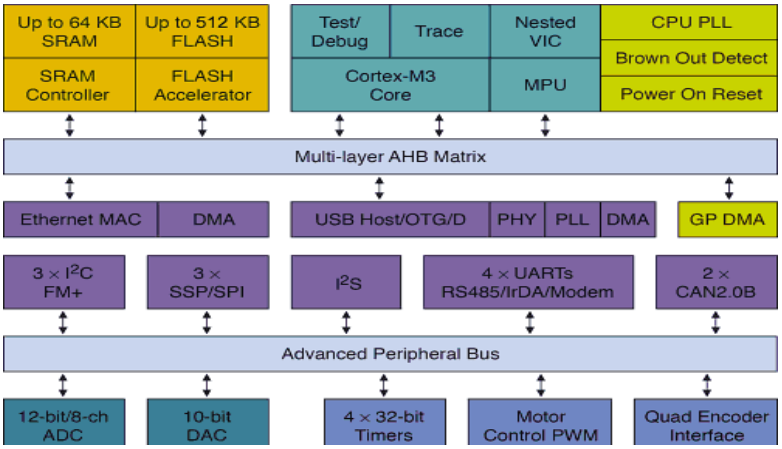

Figure 5. Functional diagram of LPC1788 /4/

3.2 LCD Panel

This section presents Liquid Crystal Display (LCD) feature and technology. It has Red Green Blue (RGB) interface with 36MHz pixel clock to show the frequency of the LCD. Its seven inch in size and comes with TFT display technology. The display mode is transmissive and resolution is 800x480 (Wide VGA size, WVGA). The number of colors is 65 kilo byte and it uses resistive touch screen technology.

Figure 6. LCD Panel /5/

4 emWin LIBRARY

“The high performance emWin embedded graphics library was developed by SEGGER. The microcontroller is now offered by NXP Semiconductors in a library form for free commercial use with the NXP microcontrollers. They are provided with an independent graphical user interface for any application that operates with a graphical LCD.” /6/ emWin is written in ANSI "C" and supports any black and white, grayscale or colour display. It needs “one of the following international standards C compilers:

- ISO/IEC/ANSI 9899:1990 (C90) with support for C++ style comments (//)
- ISO/IEC 9899:1999 (C99)
- ISO/IEC 14882:1998 (C++)” /7/

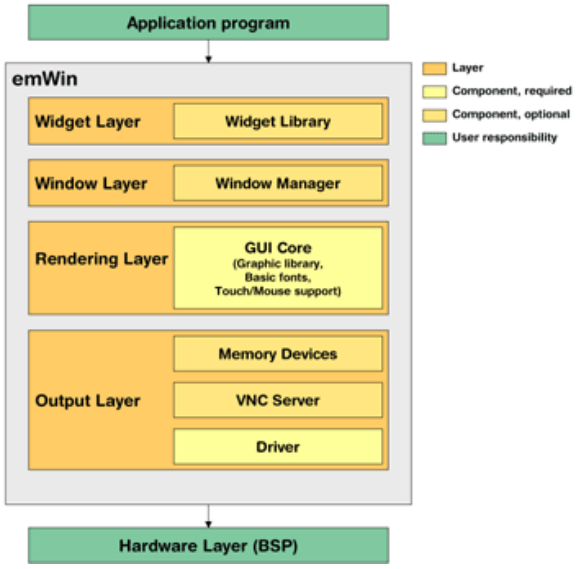

Figure 7. emWin Structure /8/

This project utility the following emWin features: Font, String/value output, window manager and Different drawing modes.

“Fonts

A variety of different fonts are shipped with the basic software: 4*6, 6*8, 6*9, 8*8, 8*9, 8*16 and my own font. /7/

String/value output routines

- Routines to show values in decimal, binary, hexadecimal, any font.
- Routines to edit values in decimal, binary, hexadecimal, any font.

Window Manager (WM)

- Complete window management including clipping.
- Callback routines supported (usage optional).
- WM uses minimum RAM (app. 50 bytes per window).” /7/

4.1 Structure of Frame Window

A frame window gives a PC application-window appearance. Figure 8 shows the detailed structure and looks of a frame window /7/

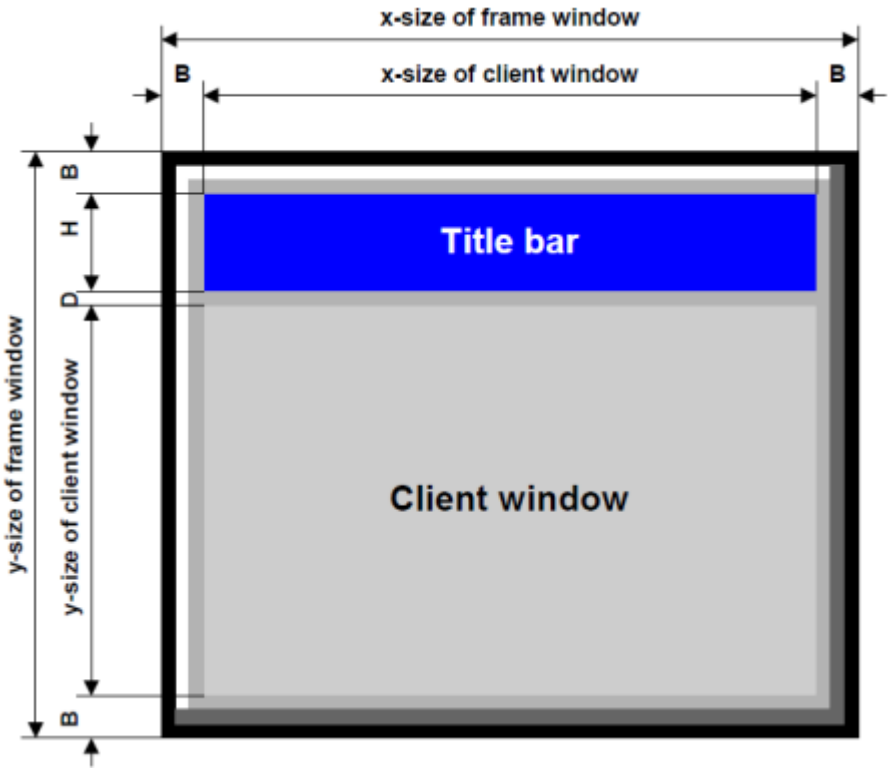

Figure 8. Sample Frame Windows /7/

4.2 Structure of Skinning API

All widget haves their own Skinning API properties of the skin, Figure 9 shows and example of one emWin widget BUTTON

Figure 9. Button skinning properties /7/

Table 1. Skinning attribute /7/

Detail	Description
A	Top color of top gradient
B	Bottom color of top gradient
C	Top color of bottom gradient.
D	Bottom color of bottom gradient
E	Outer color of surrounding frame
F	Inner color of surrounding frame
G	Color of area between surrounding frame and inner rectangular area
R	Radius of rounded corner
T	Optional text

Description
The function can be used to change the properties of the skin. /7/

Prototype

```
void BUTTON_SetSkinFlexProps(const BUTTON_SKINFLEX_PROPS * Props, int Index);
```

4.3 Dialogs

“Widgets may be created and used on their own, as they are by nature windows themselves. However, it is often desirable to use dialog boxes, which are windows that contain one or more widgets. A dialog box (or dialog) is normally a window

that appears in order to request input from the user. It may contain multiple widgets, requesting information from the user through various selections, or it may take the form of a message box which simply provides information (such as a note or warning) and an "OK" button. When a dialog created two basic things are required, such as a resource table that defines the widgets to be included, and a dialog procedure which defines the initial values for the widgets as well as their behavior.” /7/

4.3.1 **Blocking vs. non-blocking dialogs**

“Dialog windows can be blocking or non-blocking. A blocking dialog blocks the thread of execution. It has input focus by default and must be closed by the user before the thread can continue. A blocking dialog does not disable other dialogs shown at the same time. In other words, a blocking dialog is not a modal dialog. Blocking means, that the used functions (GUI_ExecDialogBox() or GUI_ExecCreatedDialog()) do not return until the dialog is closed. A non-blocking dialog, on the other hand, does not block the calling thread – it allows the task to continue while it is visible. The function returns immediately after creating the dialog. Please note that blocking functions should never be called from within callback functions.” /7/

Dialog resource table and Dialog procedure are the most constructive tools for this project and will see how they are working in at Chapter 7.

5 TOUCH USER INTERFACE SYSTEM

The touch user interface allowing the communication between the user and the home automation system are often equivalent to mainstream installation, such as home status, a temperature monitoring and a light switch.

Those systems have their own touch user interface, such as Menu frame window, Status frame window, Temperature frame window and Control frame window.

5.1 Menu Frame Window

The menu frame window displays all the needed features on the TUI and the user touch (select) a desire function. The menu frame windows consist of the following buttons:

- Status
- Temperature
- Control

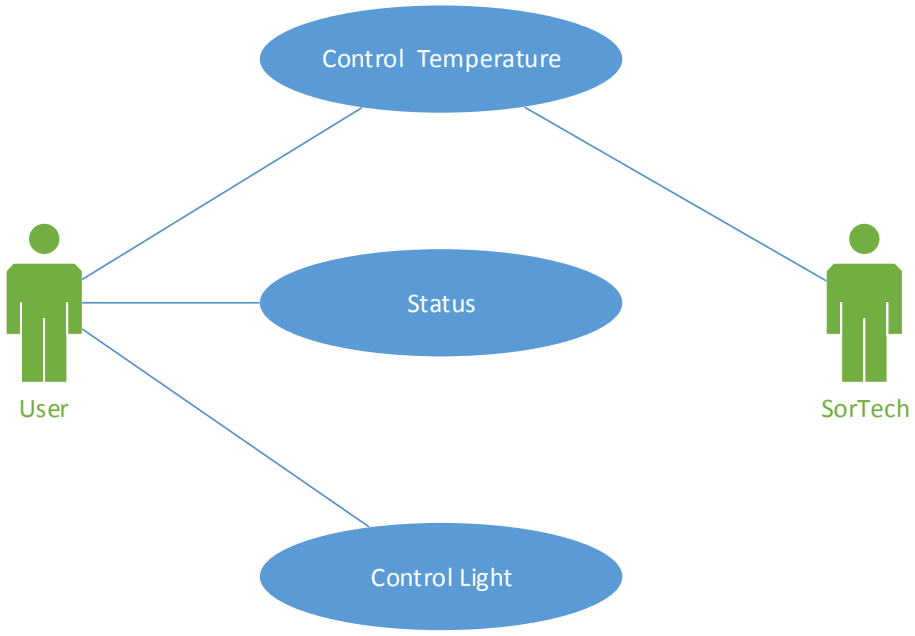

Figure 10. Use case Diagram of a home automation

5.2 Status Frame Window

The status frame window displays the current temperature of all rooms and the system informed light status (ON or OFF) of all rooms. The user requests a status window:

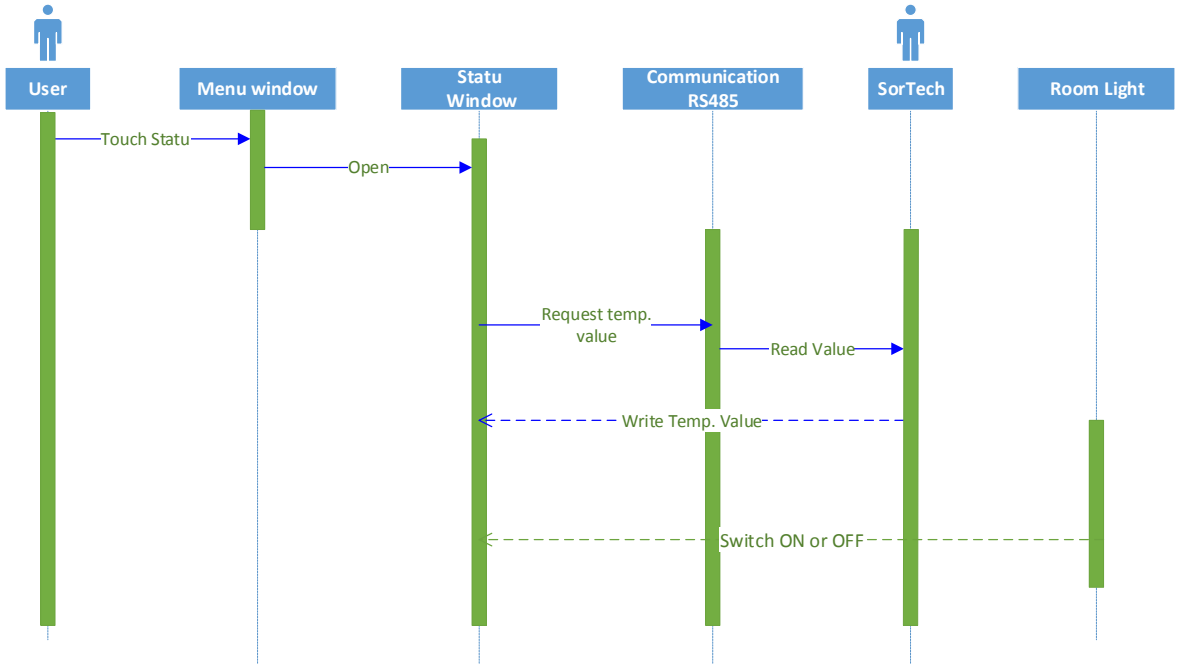

Figure 11. Sequence Diagram of a room status

5.3 Temperature Frame Window

In the temperature frame window, a user select the floor from “Floor” list box and selects the room number from the “Room Number” list box and then a system is informed of the current temperature of selected rooms and a user sets the desired temperature.

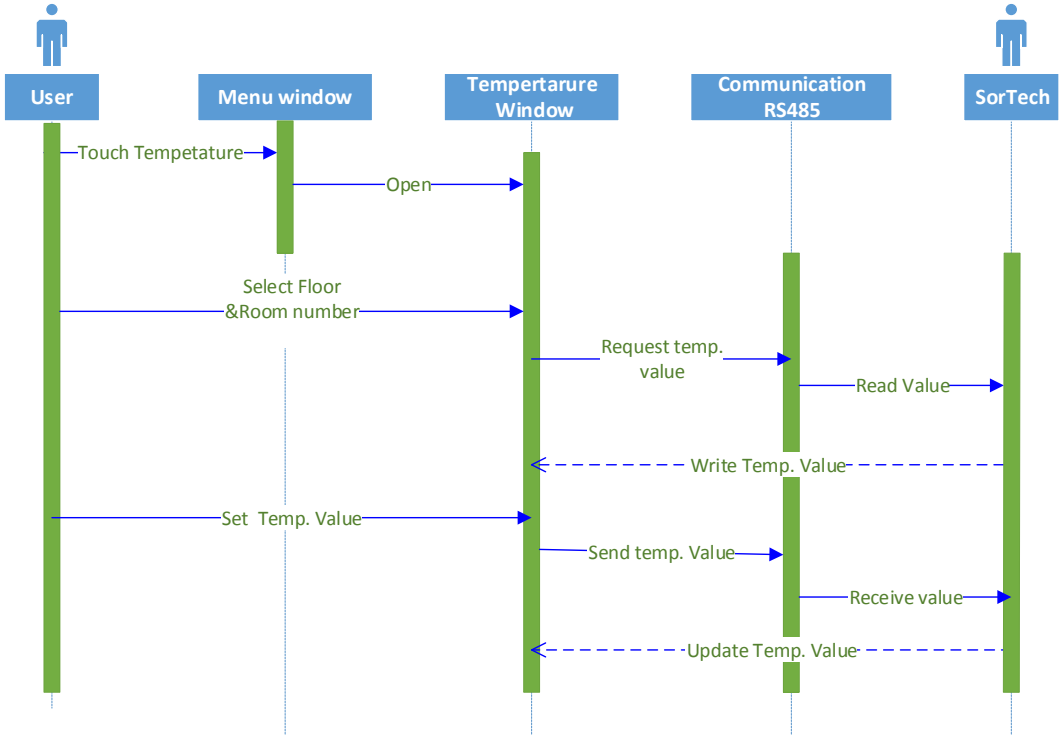

Figure 12. Sequence Diagram of room temperature monitoring and controlling

5.4 Control Frame Window

In the control Frame window, a user select the floor from the “Floor” list box and selects the room number from the “Room Number” list box and then a system is informed of the status of lamp (switch-on or switch-off) and a user can turn On or Off the of a selected room.

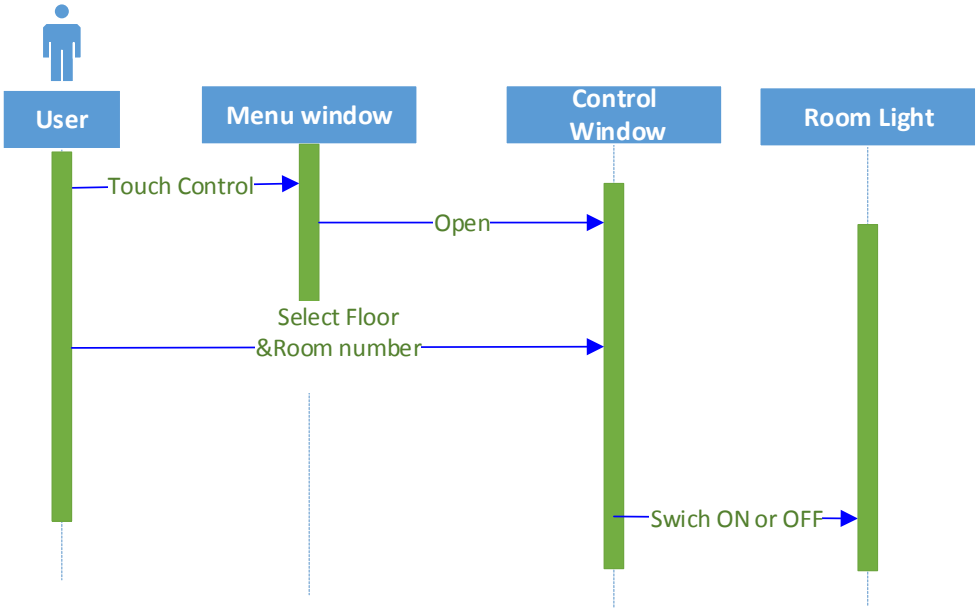

Figure 13. Sequence Diagram of a controlling room lights.

5.5 Find Coordinate Algorithm

When we draw shapes (rectangles, circles, pies, etc.), it is required to use specific positions of x and y coordinates. Simple creations can be produced by using the x and y coordinates to calculate its position. However, when we desire to draw another shape, it is not easy to put the coordinates easily. By using the Find Coordinate Algorithm derived from the Law of Sin and applied in all "frame windows", for example the "status window", shapes were able to be drawn. Figure 9 shows Law of Sin /9/ and x and y coordinates.

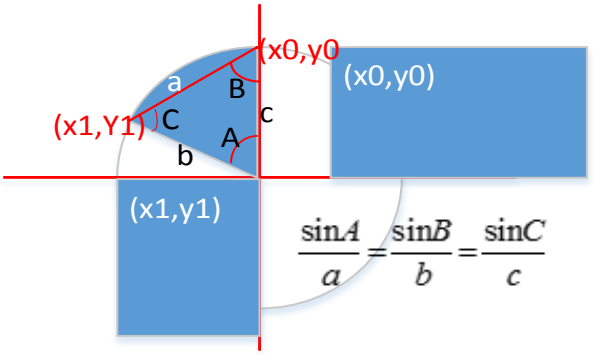

Figure 14. Law of Sine

6 COMMUNICATION

6.1 VBus Protocol

The VBus protocol, developed by RESOL, is used by a solar controller such as EL1, MIDI Pro, and DeltaSol PRO for communication purposes. It is also used to build the communication protocol in this development kit.

The first 10 byte is a header, which begins with a unique sync word 0xAA, the following header n data famers are sent, each with 6 byte in length./10/

The most important aspect of the V-Bus protocol is the special meaning attached to MSB every byte (most significant bit). This bit must always be reset with only one exception: a special synchronization byte that is used to initiate a new communication which has the value 0xAA. If a byte with set MSB is received during the communication cycle, the ongoing reception is canceled and all received data is discarded.

Table 2. General structure of Vbus protocol /10/

Synchronization byte
Header
frame 1
frame 2
…….

Header

The header contains source, destination, protocol version, command, frames count and header CRC.

Table 3. Structure of VBus header /10/

destination address (low byte)
destination address (high byte)
source address (low byte)
source address (high byte)
protocol version
command (low byte)
command (high byte)
frame count
header CRC

Destination addresses (low- and highbyte): address of the module which should receive the message.

Source address (low- and highbyte): address of the module which sent the message

Protocol version: version of the VBus protocol (0x10 for version 1.0 at the moment)

Command (low- and highbyte): device-dependent command
Frames count: number of frames attached to this header
Header CRC: checksum for integrity verification; is computed by binary inverting the sum of all header bytes, beginning with the destination address (the MSB is cleared)

Frame

Table 4. Structure of VBus frame /10/

frame byte 1
frame byte 2
frame byte 3
frame byte 4
septet

Frame byte 1: contains the first data byte; the MSB is stored in bit 0 of the septet byte

Frame byte 2: contains the second data byte; the MSB is stored in bit 1 of the septet byte

Frame byte 3: contains the third data byte; the MSB is stored in bit 2 of the septet byte

Frame byte 4: contains the fourth data byte; the MSB is stored in bit 3 of the septet byte

Septet: storage for data byte MSBs

Frame CRC: checksum for integrity verification; is computed by inverting the sum of all frame bytes (MSB is cleared).

6.2 RS485 Serial Port Interface

RS485 is one of most effective commutation protocol for industrial and home automation proposes.

Table 5. Advantage and Disadvantage of RS485 and RS232 /11/

	RS485	**RS232**
Cabling	multi-drop	single ended
Number of Devices	32 transmit 32 receive	1 transmitters 1 receivers
Communication Mode	half duplex	full duplex
Max. Distance	4000 feet at 100 Kbps	50 feet at 19.2 Kbps
Max. Data Rate	10 Mpbs for 50 feet	19.2 Kbps for 50 feet

This project uses 2 standard lines Rx and Tx for data transmission and 1 line for GND connection. It instated of SorTech cooling system.

The data are received using the asynchronous serial (COM) port of a PC. The serial port parameters are: 9600 baud, 1 start, 8 data and 1 stop bit, no parity and no handshake.

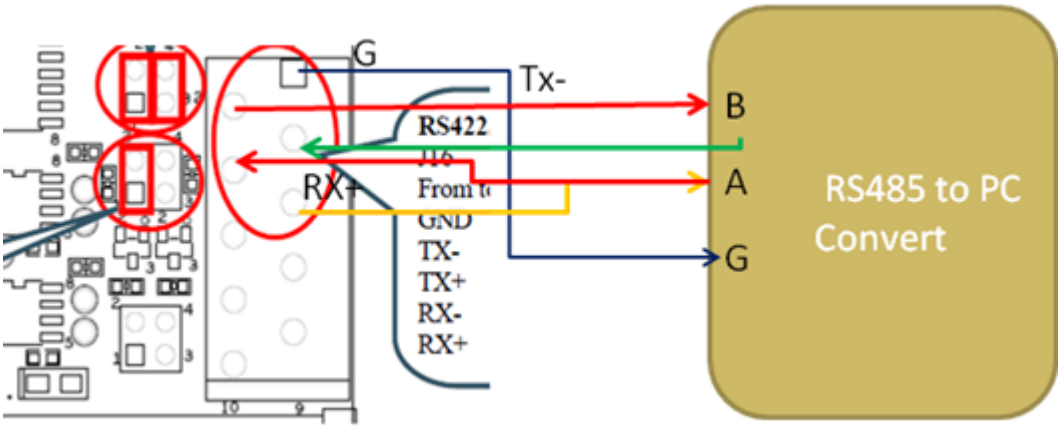

Figure 15. RS485 wiring.

7 IMPLEMENTATION

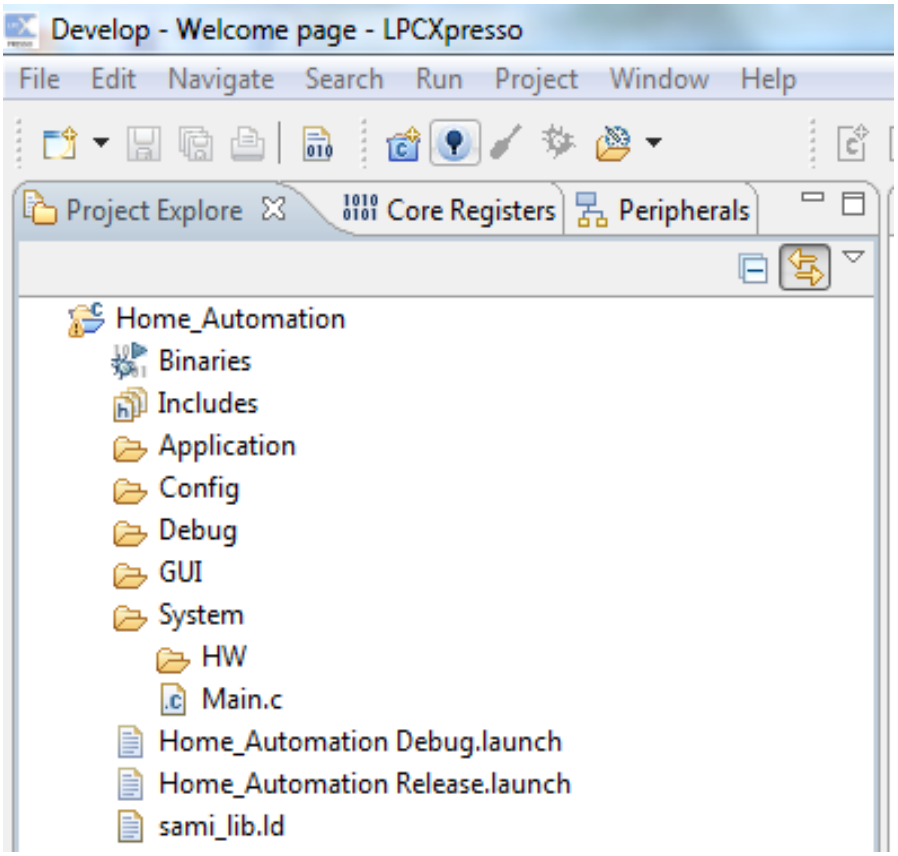

Figure 16. File directory structure

The following is a description of the folders in the Project Explorer tab:

- Application: This folder contains a mani.c, window.c, VAMK_Logo.c, count.c and cr_startup_lpc178x.c
- Config: This folder contains vBus.h, vBus.c, emWin configuration files and customized fonts.
- GUI: This folder contains all emWin header files and emWin pre-compiled libraries.
- System: This folder contains HW folder and Main.c the HW folder are contains all hardware specific functions including lm75.h,lm75.c and DeviceSupport folder.

Start-up Window

The start-up window is displayed when the application starts. This start-up window is a streamed bitmap image. Below, the figure shows the structure of the start-up window.

Figure 17. Start-up window (VAMK Logo)

`GUI_DrawGradientH()`: This function draws a rectangle filled with horizontal color gradient as background. A value of x and y is specified the size of a LCD.

```
GUI_DrawGradientH(0, 0, LCD_X_SIZE, LCD_Y_SIZE, GUI_WHITE, GUI_WHITE);
```

Figure 18. Rectangle drawing tool

```
GUI_DrawBitmapMag(&bmVAMK_Logo, 200, 130, magx, magy);
```

Figure 19. Drawing a stream bitmap.

After 5 second a window will be disappear and then a new window will be create.

```
GUI_Delay(5000);
GUI_Clear();
```

Figure 20. Clear LCD and Delay function

Main Window

After 5 seconds, the start-up window closes down and the main window opens. The main window has three elements: The menu frame window, room temperature value string and the time value string. The following figure shows the structure of the main window.

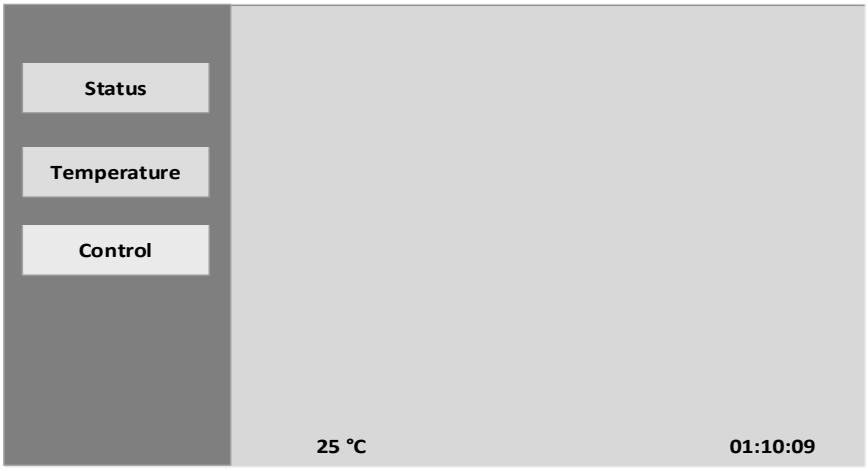

Figure 21. Main windows.

```
static void _mainWindow(WM_MESSAGE * pMsg) {
   int xSize, ySize;
   if(!flag){
       C_Temp=(float)lm75a_readTemp()/100;/*read temperature from
                                          LM75 Temperature sensor*/
       flag=TRUE;
   }

   switch (pMsg->MsgId) {
   case WM_PAINT:
       xSize = LCD_GetXSize();
       ySize = LCD_GetYSize();
       GUI_SetColor(GUI_WHITE);
       GUI_SetTextMode(GUI_TM_TRANS);
       GUI_DrawGradientH(0, 0, xSize, ySize,0X0C0C0C0 , 0X0C0C0C0);
       GUI_SetColor(0x4A494C);
       GUI_FillRect(0, 0, 200, 480);
       GUI_SetColor(GUI_WHITE);
   default:
       WM_DefaultProc(pMsg);
   }
}
```

Figure 22. Main window dialog procedures

```
static const GUI_WIDGET_CREATE_INFO _aDialogMenu[] = {
{ FRAMEWIN_CreateIndirect,"Menu"       ,     0,             10, 50, 290, 250, 0 },
{ BUTTON_CreateIndirect,  "Status     ",   GUI_ID_BUTTON0, 20, 10, 260,  35, 0 },
{ BUTTON_CreateIndirect,  "Temperature",   GUI_ID_BUTTON1, 20, 60, 260,  35, 0 },
{ BUTTON_CreateIndirect,  "Control    ",   GUI_ID_BUTTON2, 20,110, 260,  35, 0 }
};
```

Figure 23. Creation of a resource table; it contains all widgets included dialog (frame window).

```
GUI_CreateDialogBox(_aDialogMenu, GUI_COUNTOF(_aDialogMenu), &_DialogMenu, 0, 0, 0);
```

Figure 24. Menu window displayed with the above lines of code

```
_countTimer();//time counter function
GUI_SetFont(&GUI_FontRounded33);//set font type
GUI_DispStringAt(" ", 280, 440);//set a position of a string
GUI_DispFloat (C_Temp,5);//current temperature its return float value
GUI_DispString(" °C");
```

Figure 25. A main window writes a value of temperature and time.

When a user pressed a “status” button automatically “status window” is displayed the following lines of code.

```
case GUI_ID_BUTTON0: //STATUS
    WM_DeleteWindow(hDlg);
    hDlg = GUI_CreateDialogBox(_aDialogStatus, GUI_COUNTOF
    (_aDialogStatus), &_DialogStatus, WM_HBKWIN, 0, 0);
    hItem = WM_GetDialogItem(pMsg->hWin, GUI_ID_BUTTON0);
    WM_DisableWindow(hItem);

    hItem = WM_GetDialogItem(pMsg->hWin, GUI_ID_BUTTON1);
    WM_EnableWindow(hItem);
    hItem = WM_GetDialogItem(pMsg->hWin, GUI_ID_BUTTON2);
    WM_EnableWindow(hItem);
```

Figure 26. Sample dialog (window) procedure.

Status Frame

When the status button is pressed, a status window will be displayed and other windows, such as control window or temperature window disappear automatically. The following figure shows the structure of the status window.

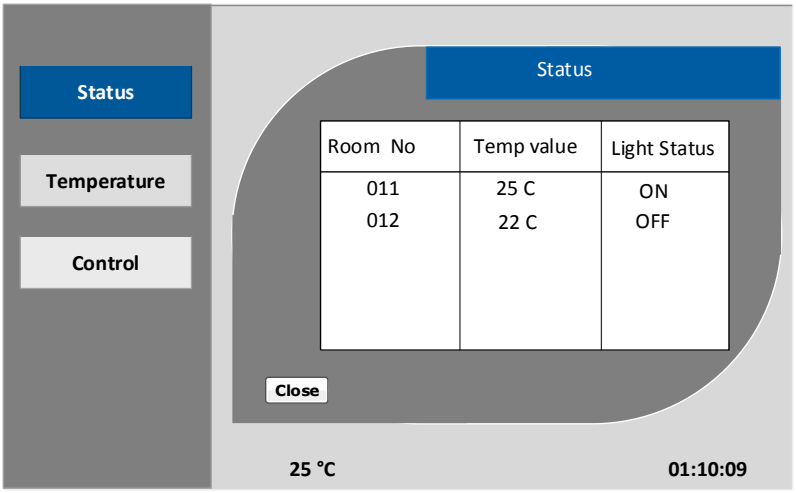

Figure 27. Status window

```
hWin = pMsg->hWin;
switch (pMsg->MsgId) {

case WM_INIT_DIALOG:
    _InitDialogStatus(hWin);
    hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_0);
    TEXT_SetText(hItem, "Status");
    TEXT_SetTextColor(hItem,GUI_WHITE);
    TEXT_SetFont(hItem,&GUI_Font24B_ASCII);

    break;
case WM_PAINT:
    _paint();
    break;
```

Figure 28. Status dialog (window) behavior.

Temperature Frame Window

When the Temperature button is pressed, a temperature window will be displayed and other windows such as control window or status window disappear automatically. The following Figure shows the structure of the temperature window.

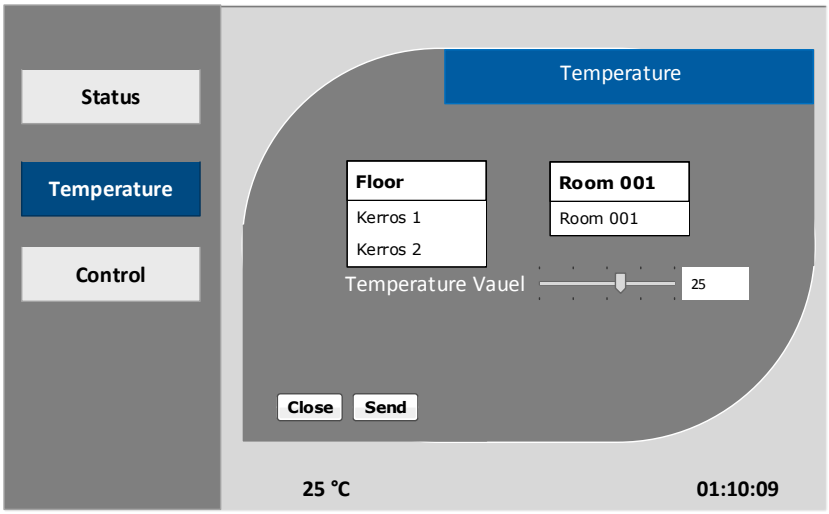

Figure 29. Temperature frame window

```
int Id = WM_GetId(pMsg->hWinSrc);
switch (Id) {
case GUI_ID_DROPDOWN0://Floor drop down  box
    hItem = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN0);
    hItemR = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN1);
    WM_ShowWindow(hItemR);
    __FeedDropDownListRoom(hItem,hItemR);
```

Figure 30. Temperature dialog (window) behavior.

When a user presses a “send” button a value of temperature is printed in this window and sent passing through RS485.The following lines of code show this task.

```
case GUI_ID_BUTTON4://Send Button
    hItem = WM_GetDialogItem(hWin, GUI_ID_EDIT0);
    read_Temp = EDIT_GetValue(hItem);
    itoa(read_Temp, buffer, 10);
    .
    .
    UART_Send(UART_2,buffer, 6, BLOCKING);
    //sending data passing through RS485
    GUI_DispStringAt(" ", 350, 440);
    GUI_DispFloat(read_Temp,5);

    GUI_DispString("°C  ");
    GUI_DispFloat(room_number,3);
```

Figure 31. Sending event

```
static void __FeedDropDownListRoom(WM_HWIN hItem,WM_HWIN hItemR){
    length = DROPDOWN_GetNumItems(hItemR);

    for(i = 0;i < length ; i++)
        DROPDOWN_DeleteItem(hItemR,0);

    DROPDOWN_SetSel(hItemR,0);
    DROPDOWN_SetBkColor(hItemR, 1, GUI_WHITE);
    DROPDOWN_SetTextColor(hItemR, 1, GUI_BLACK);
    if(DROPDOWN_GetSel(hItem)==1){//floor 1

        DROPDOWN_InsertString(hItemR, "select--",0);
        for(i=0,j=1; i < 8;i++,j++ )
            DROPDOWN_InsertString(hItemR, _Rooms[i],j);

        DROPDOWN_SetAutoScroll(hItemR,0);
    }
    else if(DROPDOWN_GetSel(hItem)==2){//floor 2

        DROPDOWN_InsertString(hItemR, "select--",0);
        for(i=8 ,j=1; i < 14;i++ , j++)
            DROPDOWN_InsertString(hItemR, _Rooms[i],j);
    }
    else if(DROPDOWN_GetSel(hItem)==3){//floor 3
```

Figure 32. Filling a room number text value into dropdown list box.

Control Frame

When the Control button is pressed, a control window will be displayed and other windows such as Status windows or Temperature windows disappear automatically. The following figure shows the structure of the control window.

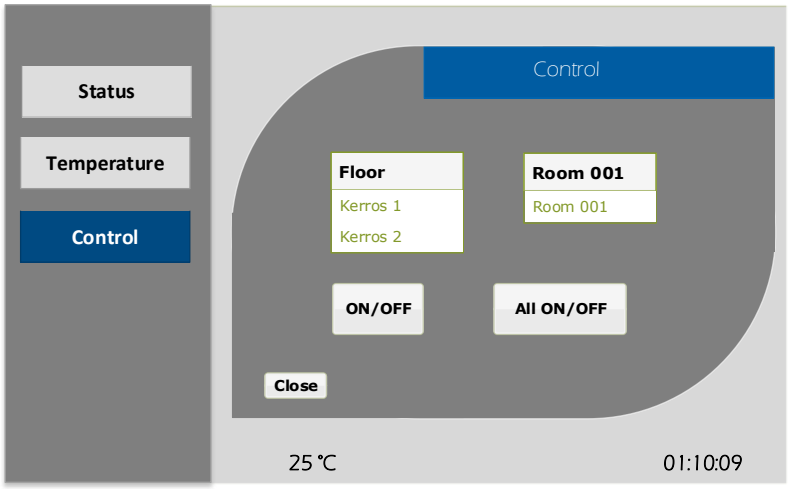

Figure 33. Control window

Painting Frame

`static void paint()`, function paints and draws all frame windows and in this case has the following functions `FindCoordinate(a0 ,a1);` and drawing emWin prototype, such as GUI_DrawPie**()**, GUI_SetColor**()** ,GUI_FillRect**().**

```
static void _paint(){
   GUI_DrawPie(xPos, yPos, r, 90, 126, 0);
   coord = FindCoordinate(a0 ,a1);
   GUI_FillRect(xPos - coord.x, yPos - coord.y, xPos + r, yPos + r);

   coord = FindCoordinate(a2 ,a3);
   GUI_FillRect(xPos2 - r,yPos2 - r , xPos2 + coord.x, yPos2 + coord.y);
```

Figure 34. Sample code paint function.

`FindCoordinate(a0 ,a1);`This function is calculating coordinate of a given angle.

```
coord.x  = (int) (sin(b * M_PI/180)/sin(c * M_PI/180)*r);
coord.y = (int) (sin(a * M_PI/180)/sin(c * M_PI/180)*r);
```

Figure 35. Sample code sine law function

```
//RS485

  UART1_RS485_CTRLCFG_Type UARTRS485ConfigStruct;
  //UART_FIFO_CFG_Type UARTFIFOConfigStruct;
  UART_CFG_Type UARTConfigStruct;

 UART_ConfigStructInit(&UARTConfigStruct);
  UARTConfigStruct.Baud_rate = 9600;

  // Initialize UART2 peripheral with given to corresponding parameter
  UART_Init(UART_2, &UARTConfigStruct);


  UART_RS485_ConfigStructInit(&UARTRS485ConfigStruct);
  UART_RS485Config(UART_2, &UARTRS485ConfigStruct);
  // Enable UART Transmit
  UART_TxCmd(UART_2, ENABLE);
```

Figure 36. Sample codes, RS485 configuration.

8 TESTING

8.1 Testing Environment

This Home Automation System used the following development and testing tools; Free Edition LPCXprsso4 IDE, RedProb+ debugging probe, LPC1788 DVK and Agilent oscilloscope.

Free Edition LPCXpresso has a limited debugging size with a limitation size of 128kb. However, it is a low-cost development tool platform available at NXP MUCs. The software consists of an enhanced, Eclipse-based IDE, a GNU C compiler, linker, libraries, and an enhanced GDB debugger. /12/

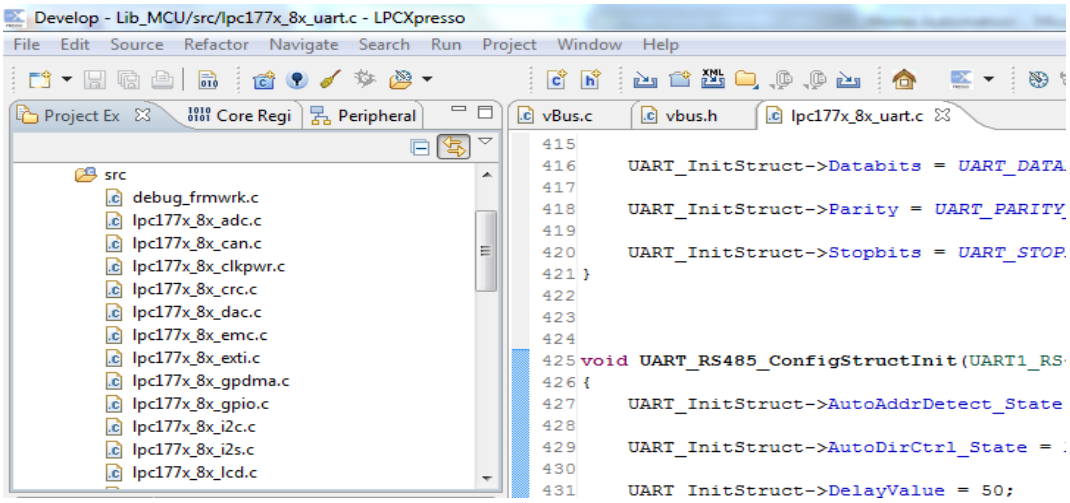

Figure 37. Development tool platforms

8.2 Results

The user interface controls home automation which are room temperature and light control. When a user presses buttons on the interface, displays of status, temperature, and light control appear. The temperature window displays, consecutively, a selection of floor numbers and room numbers followed by desired degree of temperature. Figures 21, 27, 29 and 33 show testing results.

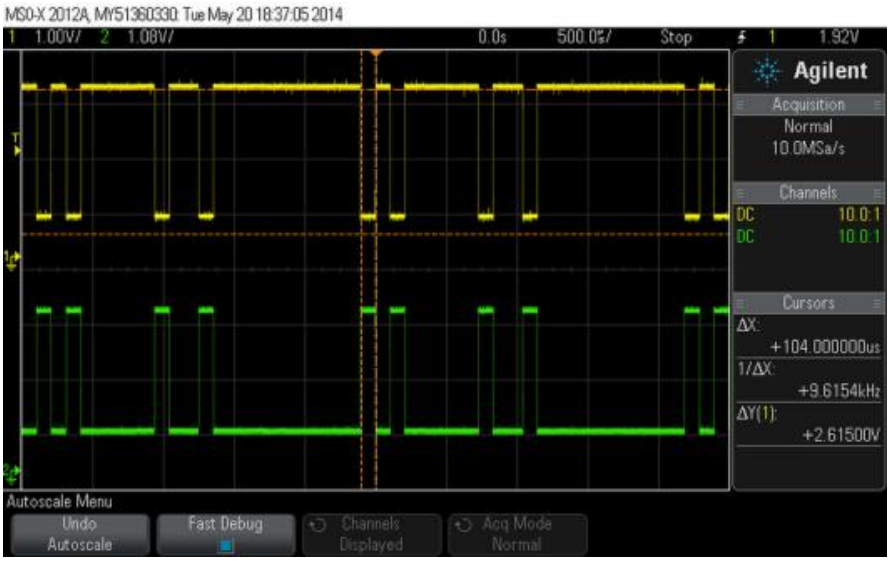

Figure 38. Oscilloscope screenshot

The RS485 serial port is transmits the desired instructions from the development kit to the testing PC. The tests carried out proved that the frequency sent by the serial port (RS485) was similar to the frequency measured as expected output. The above figure shows a result of frequency measurement.

9 CONCLUSIONS

This thesis accomplished two tasks. The Touch User Interface with its full functionality has had its design and coding completed. However, all touch user interface were tested test separately because of debugging limitation.

The communication protocol that transmits commands between the development kit and the SorTech Cooling System is partially completed. Instead of this, the development kit is connected to a desktop computer which is used to launch Putty software as a communication application to execute the sample data sent to it.

In the future, the development kit can be used as a medium (centrally) to connect smart phones and PCs with home automation systems with the objective of using the applications and web browsers on the phones to execute commands from the automation devises.

REFERENCES

/1/ Smart home. Accessed 11.06.2014. http://en.wikipedia.org/wiki/Smart_home

/2/ SorTech Adsorption Cooling System, August 2009, Design manual heat pump_ACS_V1.5_1.pdf.

/3/ NXP. Accessed 29.04.2014. http://www.nxp.com/products/microcontrollers/cortex_m3/LPC1788FET208.html

/4/ NXP. Accessed 29.04.2014. http://www.nxp.com/products/microcontrollers/cortex_m3/series/LPC1700.html

/5/ Embedded Artists web pages. Accessed 29.04.2014. http://www.embeddedartists.com/products/displays/lcdb_70.php

/6/ Segger. Accessed 20.05.2014. http://www.lpcware.com/content/project/emwin-graphics-library

/7/ Segger. Accessed 20.05.2014. http://www.segger.com/admin/uploads/productDocs/UM03001_emWin5.pdf

/8/ Segger. Accessed 20.05.2014. http://www.segger.com/emwin.html

/9/ Mathsisfun . Accessed 27.05.2014.http://www.mathsisfun.com/algebra/trig-sine-law.html

/10/ RESOL VBus Protocol Specification. 11.10.2007. VBus_Protokol_en_20071218.pdf.

/11/ Ad-net. Accessed 21.05.2014. http://www.ad-net.com.tw/index.php?id=62

/12/ NXP LPCXpresso .Accessed 04.06.2014.URL:http://www.lpcware.com/lpcxpresso

APPEARANCE

```
/*
 * window.c
 *
 *  Created on: 9.11.2013
 *      Author: Addis Ababa
 */
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "lpc_types.h"
#include "lm75a.h"
#include "GUI.h"
#include "BUTTON.h"
#include "DIALOG.h"
#include "DROPDOWN.h";

#include <stdlib.h>
#include "string.h"
#include "lpc_types.h"
#include "lpc177x_8x_pinsel.h"
#include "lpc177x_8x_uart.h"
#include "lpc177x_8x_clkpwr.h"


/***********Frist Pie settings******************/
#define a0    90   /*Starting angle*/
#define a1   150   /*Ending angle*/
#define r    200   /*Radius*/
#define xPos 173   /*x-position*/
#define yPos 200   /*y-position*/
/*****************************************/

/***********second  Pie settings******************/
#define a2   270   /*Starting angle*/
#define a3   330   /*Ending angle*/
#define r2    200   /*Radius*/
#define xPos2 374   /*x-position*/
#define yPos2 200   /*y-position*/
/*****************************************/

/*************Define Coordinate ***************/
typedef struct {
    int x;
    int y;
} Coordinate;

Coordinate coord;
/*****************************************/

#define M_PI 3.14159265358979323846264338327
#define ID_TEXT_0   (GUI_ID_USER + 0x0A)
#define ID_TEXT_1   (GUI_ID_USER + 0x0B)
#define ID_TEXT_2   (GUI_ID_USER + 0x0C)
#define ID_TEXT_3   (GUI_ID_USER + 0x0D)
#define ID_TEXT_4   (GUI_ID_USER + 0x0E)
```

```
#define ID_TEXT_5   (GUI_ID_USER + 0x0F)


//extern GUI_CONST_STORAGE GUI_FONT GUI_FontArialRoundedMTBold40;
extern GUI_CONST_STORAGE GUI_FONT GUI_FontRounded33;
extern GUI_CONST_STORAGE GUI_FONT GUI_FontDigit19;

/*********************************************************************
***************/
static BOOL_32 flag = FALSE;
static int Sec = 0;
static int Min = 0;
static int Hour = 0;
static int Value;
static  int length;

static int floorIndex;
static int roomIndex;
static int room_number;

static const char _Rooms[][22] = {
        {"ROOM P1_001"},
        {"ROOM P1_002"},
        {"ROOM P1_003"},
        {"ROOM P1_004"},
        {"ROOM P1_005"},
        {"ROOM P1_006"},
        {"ROOM P1_007"},
        {"ROOM P1_008"},

        {"ROOM P2_001"},
        {"ROOM P2_002"},
        {"ROOM P2_003"},
        {"ROOM P2_004"},
        {"ROOM P2_005"},
        {"ROOM P2_006"},

        {"ROOM P3_001"},
        {"ROOM P3_002"},
        {"ROOM P3_003"},
        {"ROOM P3_004"},
        {"ROOM P3_005"},

        {"ROOM PK_001"},
        {"ROOM PK_002"},
        {"ROOM PK_003"},

};
/*********************************************************************
*****************/

WM_HWIN hDlg;
float C_Temp=0;
float read_Temp;
char buffer[20];
unsigned int i,j;
```

```
/*****************************Define proto-
type*********************************/

static void _AddDropDownList(int hDropDownF);

/*******************************End proto-
type*********************************/



/*****************************************************************
******************
 *
*
 *                    _countTimer  Function
*
 *
*

*****************************************************************
******************/
static void _countTimer() {
    if (++Sec == 60) {
        Sec = 0;
        if (++Min == 60) {
            Min = 0;
            if (++Hour == 24) {
                Hour = 0;
            }
        }
    }
}
/*******************************End counTi-
mer*********************************/

/*****************************************************************
******************
 *
*
 *                     Coordinate Finder Function
*
 *
*

*****************************************************************
******************/

Coordinate FindCoordinate(int angle0,int angle1) {
    int a,b,c;  //angles

    b = angle1 - angle0;
    c = 90 ;
    a = 180 - (b + c);//radian

    coord.x  = (int) (sin(b * M_PI/180)/sin(c * M_PI/180)*r);
    coord.y = (int) (sin(a * M_PI/180)/sin(c * M_PI/180)*r);

    return coord;
```

```
}
/*********************************End coordi-
nate*************************************/

/*****************************************************************
*********************
 *
 *                   _paint Function
 *
 *
*****************************************************************
*****************/
static void _paint(){

    GUI_DrawPie(xPos, yPos, r, 90, 126, 0);
    GUI_SetColor(0x4A494C);
    GUI_DrawPie(xPos, yPos, r, a0, a1, 0);
    GUI_DrawPie(xPos2, yPos2, r2, a2, a3, 0);

    coord = FindCoordinate(a0 ,a1);
    GUI_FillRect(xPos - coord.x, yPos - coord.y, xPos + r, yPos +
r);

    coord = FindCoordinate(a2 ,a3);
    GUI_FillRect(xPos2 - r,yPos2 - r , xPos2 + coord.x, yPos2 +
coord.y);


    GUI_SetColor(GUI_BLUE);
    GUI_FillRect(173,0,547,40);
    GUI_DrawPie(xPos, yPos, r, 90, 126, 0);

    GUI_SetColor(0x4A494C);
    coord = FindCoordinate(90 ,127);
    GUI_FillRect(xPos - coord.x, yPos - coord.y, r, r);

    GUI_SetColor(GUI_WHITE);
    GUI_DrawHLine(40,55,547);
}
/***********************************End
_paint*********************************************/


/*****************************************************************
*****************************
 *
 *                              _customizeButton Function
 *
 *
*****************************************************************
************************/

void _customizeButton(int type){
    BUTTON_SKINFLEX_PROPS Props;

    if( type == 1 ){
```

```
        BUTTON_GetSkinFlexProps(&Props, BUT-
TON_SKINFLEX_PI_DISABLED);
        Props.aColorFrame[0]=GUI_BLUE;
        Props.aColorFrame[1]=GUI_BLUE;
        Props.aColorFrame[2]=GUI_BLUE;
        Props.aColorLower[0]=GUI_BLUE;
        Props.aColorLower[1]=GUI_BLUE;
        Props.aColorUpper[0]=GUI_BLUE;
        Props.aColorUpper[1]=GUI_BLUE;
        Props.Radius = 5;
        BUTTON_SetSkinFlexProps(&Props, BUT-
TON_SKINFLEX_PI_DISABLED);
        //WM_InvalidateWindow(hWin);
    }
    else if( type == 2 ){
        BUTTON_GetSkinFlexProps(&Props, 1);
        Props.aColorFrame[0]=GUI_BLUE;
        Props.aColorFrame[1]=GUI_BLUE;
        Props.aColorFrame[2]=GUI_BLUE;
        Props.aColorLower[0]=GUI_BLUE;
        Props.aColorLower[1]=GUI_BLUE;
        Props.aColorUpper[0]=GUI_BLUE;
        Props.aColorUpper[1]=GUI_BLUE;
        Props.Radius = 5;
        BUTTON_SetSkinFlexProps(&Props, 1);
    }

}
/***********************************End
_customizeButton*********************************************/


/*****************************************************************
**************************
 *
 *              Create
 *
 *
*****************************************************************
**********************/

static const GUI_WIDGET_CREATE_INFO _aDialogMenu[] = {
{ FRAMEWIN_CreateIndirect,"Menu"       ,    0,             10, 50,
190, 150, 0 },
{ BUTTON_CreateIndirect,   "Status     ",   GUI_ID_BUTTON0, 20,
10, 160,  35, 0 },
{ BUTTON_CreateIndirect,   "Temperature",   GUI_ID_BUTTON1, 20,
60, 160,  35, 0 },
{ BUTTON_CreateIndirect,   "Control    ",   GUI_ID_BUTTON2,
20,110, 160,  35, 0 }
};

static const GUI_WIDGET_CREATE_INFO _aDialogStatus[] = {
        { FRAMEWIN_CreateIndirect, "Status",  1,            227,
30,  548, 401,            0 },
        { TEXT_CreateIndirect,   "Status",ID_TEXT_0,        280,
10,  100,  36, TEXT_CF_LEFT },
```

```
        { LISTVIEW_CreateIndirect, NULL, GUI_ID_LISTVIEW0,   50,
50, 450, 250,            0 },
};


static const GUI_WIDGET_CREATE_INFO _aDialogTemp[] = {
        { FRAMEWIN_CreateIndirect,     "Temperature",
2,      227,  30, 548, 401, 0 },
        { BUTTON_CreateIndirect,     "Close",  GUI_ID_BUTTON3,
30, 360, 100,  30, 0 },
        { BUTTON_CreateIndirect,     "send",  GUI_ID_BUTTON4,
130, 360, 100,  30, 0 },
        { TEXT_CreateIndirect,        "Temperature",  ID_TEXT_1,
260, 10,150, 36, TEXT_CF_LEFT },
        { TEXT_CreateIndirect,       "Floor",  ID_TEXT_3, 100,
80,100, 36, TEXT_CF_LEFT },
        { TEXT_CreateIndirect,     NULL,  ID_TEXT_4, 250, 80,100,
36, TEXT_CF_LEFT },
        { DROPDOWN_CreateIndirect, NULL,  GUI_ID_DROPDOWN0,  95,
100, 130,  70, 0 },
        { DROPDOWN_CreateIndirect, NULL,  GUI_ID_DROPDOWN1, 245,
100, 130,  115, 0 },//80
        { TEXT_CreateIndirect,   NULL,  ID_TEXT_5, 100,  150, 105,
20, TEXT_CF_LEFT | TEXT_CF_VCENTER },
        { SLIDER_CreateIndirect,   NULL,    GUI_ID_SLIDER0, 205,
150, 145,  20 },
        { EDIT_CreateIndirect,     NULL ,    GUI_ID_EDIT0,  350,
150,  30,  20, 0, 0}
};


static const GUI_WIDGET_CREATE_INFO _aDialogControl[] = {
        { FRAMEWIN_CreateIndirect, "Control",  3,
227,  30, 548, 401, 0 },
        { BUTTON_CreateIndirect,   "Close",  GUI_ID_BUTTON5,
30, 360, 100,  30, 0 },
        { BUTTON_CreateIndirect,   "ON",  GUI_ID_BUTTON6,
110, 160, 60,  30, 0 },
        { TEXT_CreateIndirect,       "Floor",  ID_TEXT_3, 100,
80,100, 36, TEXT_CF_LEFT },
        { TEXT_CreateIndirect,     NULL,  ID_TEXT_4, 250, 80,100,
36, TEXT_CF_LEFT },
        { DROPDOWN_CreateIndirect, NULL,  GUI_ID_DROPDOWN2,  95,
100, 130,  70, 0 },
        { DROPDOWN_CreateIndirect, NULL,  GUI_ID_DROPDOWN3, 245,
100, 130,  115, 0 },//80
        { TEXT_CreateIndirect, "Control",ID_TEXT_2, 200, 10,200,
36, TEXT_CF_LEFT }
};
/*********************************End*************************
******************/


/*****************************Start
Init****************************************/
static void _InitDialogTemp(WM_HWIN hWin) {
```

```
    /* Init frame window */
    FRAMEWIN_SetBorderSize(hWin, 0);
    FRAMEWIN_SetTitleVis(hWin, 0);

    _customizeButton(1);

}

static void _InitDialogControl(WM_HWIN hWin) {

    /* Init Control frame window */
    FRAMEWIN_SetBorderSize(hWin, 0);
    FRAMEWIN_SetTitleVis(hWin, 0);
    _customizeButton(1);

}

static void _InitDialogStatus(WM_HWIN hWin) {
    /* Init Status frame window */
    FRAMEWIN_SetBorderSize(hWin, 0);
    FRAMEWIN_SetTitleVis(hWin, 0);
    _customizeButton(1);

}

static void _InitDialogMenu(WM_HWIN hWin) {
    /* Init Menu frame window*/
    FRAMEWIN_SetBorderSize(hWin, 0);
    FRAMEWIN_SetClientColor(hWin, 0x4A494C);
    FRAMEWIN_SetTitleVis(hWin, 0);
}

/*****************************End
Init*******************************************/


/*****************************************************************
**************************
 *
 *                      Floor    __AddDropDownList  Function
 *
 *
*****************************************************************
**********************/
static void _AddDropDownList(int hDropDownF ){
    WM_HWIN hWin,hItem;
    //WM_MESSAGE * pMsg;
    //hWin = pMsg->hWin;
    hItem = WM_GetDialogItem(hWin,hDropDownF );

    DROPDOWN_AddString(hItem, " Select");
    DROPDOWN_AddString(hItem, "Kerros_P1");
    DROPDOWN_AddString(hItem, "Kerros_P2");
    DROPDOWN_AddString(hItem, "Kerros_P3");
    DROPDOWN_AddString(hItem, "Kerros_PK");
    DROPDOWN_SetBkColor(hItem, 1, GUI_WHITE);
    DROPDOWN_SetTextColor(hItem, 1, GUI_BLACK);
}
```

```
/***********************************End
_AddDropDownList*********************************/

/*************************************************************************
***************************
 *
 *                                __FeedDropDownListRoom Function
 *
 *
*************************************************************************
***********************/
static void __FeedDropDownListRoom(WM_HWIN hItem,WM_HWIN hItemR){

    length = DROPDOWN_GetNumItems(hItemR);

    for(i = 0;i < length ; i++)
        DROPDOWN_DeleteItem(hItemR,0);

    DROPDOWN_SetSel(hItemR,0);
    DROPDOWN_SetBkColor(hItemR, 1, GUI_WHITE);
    DROPDOWN_SetTextColor(hItemR, 1, GUI_BLACK);

    if(DROPDOWN_GetSel(hItem)==1){//floor 1

        DROPDOWN_InsertString(hItemR, "select--",0);
        for(i=0,j=1; i < 8;i++,j++ )
            DROPDOWN_InsertString(hItemR, _Rooms[i],j);

        DROPDOWN_SetAutoScroll(hItemR,0);
    }
    else if(DROPDOWN_GetSel(hItem)==2){//floor 2

        DROPDOWN_InsertString(hItemR, "select--",0);
        for(i=8 ,j=1; i < 14;i++ , j++)
            DROPDOWN_InsertString(hItemR, _Rooms[i],j);
    }
    else if(DROPDOWN_GetSel(hItem)==3){//floor 3

        DROPDOWN_InsertString(hItemR, "select--",0);
        for(i=14 ,j=1; i < 19;i++ , j++)
            DROPDOWN_InsertString(hItemR, _Rooms[i],j);

    }
    else if(DROPDOWN_GetSel(hItem)==4){//floor k

        DROPDOWN_InsertString(hItemR, "select--",0);
        for(i=19 ,j=1; i < 22;i++ , j++)
            DROPDOWN_InsertString(hItemR, _Rooms[i],j);
    }
}
/************************End _FeedDropDownList Room
number**************************/

/*************************************************************************
****
*
*                            _AddListviewItem
```

```
*
**********************************************************************
****/
static void _AddListviewItem(LISTVIEW_Handle hObj, const char*
pRoom) {
  unsigned NumItems;
  NumItems = LISTVIEW_GetNumRows(hObj);
  LISTVIEW_AddRow(hObj, NULL);
  LISTVIEW_SetItemText(hObj, 0, NumItems, pRoom);

}
/************************End _AddListviewItem
*************************/



/**********************************************************************
******************
 *
*
 *                    Main Frame window
*
 *
*

**********************************************************************
******************/

 static void _mainWindow(WM_MESSAGE * pMsg) {
    int xSize, ySize;
    if(!flag){
        C_Temp=(float)lm75a_readTemp()/100;/*read temperature from
                                            LM75 Temperature sen-
sor*/
        flag=TRUE;
    }

    switch (pMsg->MsgId) {
    case WM_PAINT:
        xSize = LCD_GetXSize();
        ySize = LCD_GetYSize();
        GUI_SetColor(GUI_WHITE);
        GUI_DrawGradientH(0, 0, xSize, ySize,0X0C0C0C0 ,
0X0C0C0C0);
        GUI_SetColor(0x4A494C);
        GUI_FillRect(0, 0, 200, 480);
        GUI_SetColor(GUI_WHITE);
    default:
        WM_DefaultProc(pMsg);
    }
 }

/*****************************End
Main*****************************************/

/**********************************************************************
******************
```

```
 *
*
 *                      Temp Frame window
*
 *
*

**********************************************************************
******************/

static void _DialogTemp(WM_MESSAGE * pMsg) {
    WM_HWIN hWin,hItem,hItemR;


    hWin = pMsg->hWin;

    switch (pMsg->MsgId) {

    case WM_INIT_DIALOG:

        _InitDialogTemp(hWin);
        //_AddDropDownList(GUI_ID_DROPDOWN0);

        hItem = WM_GetDialogItem(hWin,GUI_ID_DROPDOWN0 );
            DROPDOWN_AddString(hItem, " Select");
            DROPDOWN_AddString(hItem, "Kerros_P1");
            DROPDOWN_AddString(hItem, "Kerros_P2");
            DROPDOWN_AddString(hItem, "Kerros_P3");
            DROPDOWN_AddString(hItem, "Kerros_PK");

        hItem = WM_GetDialogItem(hWin, GUI_ID_BUTTON4);
        WM_HideWindow(hItem);
        //DROPDOWN_SetFont(hItem,&GUI_Font24B_ASCII);

        hItem = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN1);
        WM_HideWindow(hItem);

        //DROPDOWN_SetFont(hItem,&GUI_Font24B_ASCII);
        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_5);//Label
"Temperature values"
        TEXT_SetTextColor(hItem,GUI_WHITE) ;

        hItem = WM_GetDialogItem(hWin, GUI_ID_SLIDER0);//Slider
        SLIDER_SetRange(hItem, -18, 40);
        SLIDER_SetValue(hItem, C_Temp);
        WM_HideWindow(hItem);

        hItem = WM_GetDialogItem(hWin, GUI_ID_EDIT0);//Text Box
"values"
        EDIT_SetDecMode(hItem, Value, -18, 40, 0, 1);
        WM_DisableWindow(hItem);
        WM_HideWindow(hItem);


        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_3);//Label
Floor
        TEXT_SetTextColor(hItem,GUI_WHITE);
```

```
        TEXT_SetFont(hItem,&GUI_Font16B_ASCII);

        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_4);//Label
Room Number
        TEXT_SetTextColor(hItem,GUI_WHITE);
        TEXT_SetFont(hItem,&GUI_Font16B_ASCII);

        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_1);//Label
Temperature on frame
        TEXT_SetTextColor(hItem,GUI_WHITE);
        TEXT_SetFont(hItem,&GUI_Font24B_ASCII);


        break;
    case WM_PAINT:
        _paint();

        break;
    case WM_NOTIFY_PARENT:

        if (pMsg->Data.v == WM_NOTIFICATION_RELEASED) {
            int Id = WM_GetId(pMsg->hWinSrc);
            switch (Id) {
            case GUI_ID_BUTTON3:// Close Button
                    GUI_EndDialog(hWin, 0);
            break;
            case GUI_ID_BUTTON4://Send Button
                hItem = WM_GetDialogItem(hWin, GUI_ID_EDIT0);
                read_Temp = EDIT_GetValue(hItem);
                //EDIT_GetText(hItem,buffer,5);
                itoa(read_Temp, buffer, 10);
                // C_Temp = atoi(buffer);


                  hItem = WM_GetDialogItem(hWin,
GUI_ID_DROPDOWN0);
                  floorIndex = DROPDOWN_GetSel(hItem);

                  hItemR = WM_GetDialogItem(hWin,
GUI_ID_DROPDOWN1);
                  roomIndex = DROPDOWN_GetSel(hItemR);

                      switch (floorIndex) {
                              case 1:
                              switch (roomIndex) {
                                      case 1: room_number=11;
break;
                                      case 2: room_number=12;
break;
                                      case 3: room_number=13;
break;
                                      case 4: room_number=14;
break;
                                      case 5: room_number=15;
break;
                                      case 6: room_number=16;
break;
```

```
                                        case 7: room_number=17;
break;
                                        case 8: room_number=18;
break;
                                        }
                                break;
                                case 2:
                                    switch (roomIndex) {
                                            case 1: room_number=21;
break;
                                            case 2: room_number=22;
break;
                                            case 3: room_number=23;
break;
                                            case 4: room_number=24;
break;
                                            case 5: room_number=25;
break;
                                            case 6: room_number=26;
break;
                                            }
                                break;
                                case 3:
                                switch (roomIndex) {
                                            case 1: room_number=31;
break;
                                            case 2: room_number=32;
break;
                                            case 3: room_number=33;
break;
                                            case 4: room_number=34;
break;
                                            case 5: room_number=35;
break;
                                            }
                                break;
                                case 4:
                                    switch (roomIndex) {
                                            case 1: room_number=01;
break;
                                            case 2: room_number=02;
break;
                                            case 3: room_number=03;
break;
                                            }
                                  break;
                                }
                        UART_Send(UART_2,buffer, 6, BLOCKING);
                        GUI_SetFont(&GUI_FontRounded33);
                                    GUI_DispStringAt(" ", 400, 440);
                                    GUI_DispFloat(read_Temp,5);

                                    GUI_DispString("°C  ");
                                    GUI_DispFloat(room_number,3);
                                    //UART_RS485SendData(UART_2,
buffer, sizeof(buffer));
```

```
                // C_Temp1--;
                //GUI_DispCEOL();
                /*  GUI_SetFont(&GUI_FontRounded33);
          GUI_DispStringAt(" ", 300, 440);*/
                //GUI_DispFloat (C_Temp1,5);
                //GUI_SetFont(&GUI_FontArialRoundedMTBold50);
                //GUI_DispString("°C");
                break;
            }
        }
        if (pMsg->Data.v == WM_NOTIFICATION_SEL_CHANGED){//||
pMsg->Data.v ==WM_NOTIFICATION_CLICKED) {

            hItem =WM_GetDialogItem(pMsg->hWin, ID_TEXT_4);// La-
bel Room Number
            TEXT_SetText(hItem,"Room Number");

            int Id = WM_GetId(pMsg->hWinSrc);
            switch (Id) {
            case GUI_ID_DROPDOWN0://Floor drop down  box
                hItem = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN0);
                hItemR = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN1);
                WM_ShowWindow(hItemR);
                __FeedDropDownListRoom(hItem,hItemR);

                break;
            case GUI_ID_DROPDOWN1://Room number drop down box

                hItem =WM_GetDialogItem(pMsg->hWin,
ID_TEXT_5);//Label Temperature values
                TEXT_SetText(hItem,"Temperature values :");
                hItem = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN1);

                if(DROPDOWN_GetSel(hItem)!=0){
                    hItem = WM_GetDialogItem(hWin,
GUI_ID_SLIDER0);
                    WM_ShowWindow(hItem);
                    hItem = WM_GetDialogItem(hWin, GUI_ID_EDIT0);
                    WM_ShowWindow(hItem);
                    hItem = WM_GetDialogItem(hWin,
GUI_ID_BUTTON4);
                            WM_ShowWindow(hItem);

                }

                break;
            }

        }
        if (pMsg->Data.v == WM_NOTIFICATION_VALUE_CHANGED) {
            hItem = WM_GetDialogItem(hWin, GUI_ID_SLIDER0);
            Value = SLIDER_GetValue(hItem);
            hItem = WM_GetDialogItem(hWin, GUI_ID_EDIT0);
            EDIT_SetValue(hItem, Value);
        }

        break;
```

```
    default:
        WM_DefaultProc(pMsg);
    }
}
/*****************************End
temp*************************************/

/*****************************************************************
******************
 *
*
 *                    Control Frame window
*
 *
*

*****************************************************************
*****************/

static void _DialogControl(WM_MESSAGE * pMsg) {
    WM_HWIN hWin,hItem,hItemR;

    hWin = pMsg->hWin;
    switch (pMsg->MsgId) {

    case WM_INIT_DIALOG:
        _InitDialogControl(hWin);
        //_AddDropDownList(GUI_ID_DROPDOWN2);

        hItem = WM_GetDialogItem(hWin,GUI_ID_DROPDOWN2 );
            DROPDOWN_AddString(hItem, " Select");
            DROPDOWN_AddString(hItem, "Kerros_P1");
            DROPDOWN_AddString(hItem, "Kerros_P2");
            DROPDOWN_AddString(hItem, "Kerros_P3");
            DROPDOWN_AddString(hItem, "Kerros_PK");

        hItem = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN3);
        WM_HideWindow(hItem);

        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_3);
        TEXT_SetTextColor(hItem,GUI_WHITE);
        TEXT_SetFont(hItem,&GUI_Font16B_ASCII);

        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_4);
        TEXT_SetTextColor(hItem,GUI_WHITE);
        TEXT_SetFont(hItem,&GUI_Font16B_ASCII);


        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_2);
        TEXT_SetText(hItem, "Lighting Control");
        TEXT_SetTextColor(hItem,GUI_WHITE);
        TEXT_SetFont(hItem,&GUI_Font24B_ASCII);
        break;
    case WM_PAINT:
        _paint();

        break;
```

```
    case WM_NOTIFY_PARENT:
        if (pMsg->Data.v == WM_NOTIFICATION_RELEASED) {
            int Id = WM_GetId(pMsg->hWinSrc);
            switch (Id) {
            case GUI_ID_BUTTON5:
                GUI_EndDialog(hWin, 0);
                break;
            case GUI_ID_BUTTON6:
                /*  hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON6);
                    WM_DisableWindow(hItem);*/
                /*  C_Temp1--;
          if(C_Temp1 < 20)
              GUI_SetColor(GUI_WHITE);
          else if(C_Temp1 > 30)

              GUI_SetColor(GUI_RED);
          else if(C_Temp1 >= 20 && C_Temp1 <= 30)
              GUI_SetColor(GUI_BLUE);
          GUI_SetFont(&GUI_FontRounded33);
          GUI_DispStringAt(" ", 280, 440);
          GUI_DispFloat (C_Temp1,5);*/
                // GUI DispCEOL();

                break;
            }
        }
        if (pMsg->Data.v == WM_NOTIFICATION_SEL_CHANGED){

            hItem =WM_GetDialogItem(pMsg->hWin, ID_TEXT_4);
            TEXT_SetText(hItem,"Room Number");
            int Id = WM_GetId(pMsg->hWinSrc);
            switch (Id) {
            case GUI_ID_DROPDOWN2://Floor drop down  box
                hItem = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN2);
                hItemR = WM_GetDialogItem(hWin, GUI_ID_DROPDOWN3);
                WM_ShowWindow(hItemR);
                __FeedDropDownListRoom(hItem, hItemR);
                break;
            case GUI_ID_DROPDOWN3://Room number drop down box

                break;
            }

        }

        break;
    default:
        WM_DefaultProc(pMsg);
    }
}
/*****************************End
_DialogControl*****************************************/
```

```
/*********************************************************************
******************
 *
*
 *                     Status Frame window
*
 *
*

**********************************************************************
******************/

static void _DialogStatus(WM_MESSAGE * pMsg) {
    WM_HWIN hWin,hItem;

    hWin = pMsg->hWin;
    switch (pMsg->MsgId) {

    case WM_INIT_DIALOG:
        _InitDialogStatus(hWin);
        hItem = WM_GetDialogItem(pMsg->hWin, ID_TEXT_0);
        TEXT_SetText(hItem, "Status");
        TEXT_SetTextColor(hItem,GUI_WHITE);
        TEXT_SetFont(hItem,&GUI_Font24B_ASCII);

        hItem = WM_GetDialogItem(hWin, GUI_ID_LISTVIEW0);
        WM_SetScrollbarV(hItem, 1);

        LISTVIEW_SetGridVis(hItem, 1);
        LISTVIEW_SetLBorder(hItem, 50);
        LISTVIEW_SetRBorder(hItem, 10);
        LISTVIEW_SetRowHeight(hItem,30);
        LISTVIEW_SetBkColor(hItem,0,0x4A494C);
        LISTVIEW_AddColumn(hItem,150,"Room Number",2);
        LISTVIEW_AddColumn(hItem,150,"Temperature",2);
        LISTVIEW_AddColumn(hItem,150,"Light",2);

//_Rooms[0][i]
          while (i < 22) {
            _AddListviewItem(hItem, _Rooms[i]);
            i++;
          }
        break;
    case WM_PAINT:
        _paint();
        break;
    case WM_NOTIFY_PARENT:
        if (pMsg->Data.v == WM_NOTIFICATION_RELEASED) {
            int Id = WM_GetId(pMsg->hWinSrc);
            switch (Id) {
            /*   case GUI_ID_OK:
          break;*/
            case GUI_ID_CANCEL:

                GUI_EndDialog(hWin, 0);
                break;
            }
```

```
        }
        break;
    default:
        WM_DefaultProc(pMsg);
    }
}
/*****************************End
Status**************************************************/


/*****************************************************************
*************************
 *
*
 *                    Menu Frame window
*
 *
*

*****************************************************************
************************/
static void _DialogMenu(WM_MESSAGE * pMsg) {
    WM_HWIN hWin,hItem;

    hWin = pMsg->hWin;

    switch (pMsg->MsgId) {
    case WM_INIT_DIALOG:
        _InitDialogMenu(hWin);
        hItem = WM_GetDialogItem(pMsg->hWin, GUI_ID_BUTTON0);
        BUTTON_SetTextColor(hItem,2,GUI_WHITE);

        hItem = WM_GetDialogItem(pMsg->hWin, GUI_ID_BUTTON1);
        BUTTON_SetTextColor(hItem,2,GUI_WHITE);

        hItem = WM_GetDialogItem(pMsg->hWin, GUI_ID_BUTTON2);
        BUTTON_SetTextColor(hItem,2,GUI_WHITE);

        break;
    case WM_NOTIFY_PARENT:
        if (pMsg->Data.v == WM_NOTIFICATION_RELEASED) {
            int Id = WM_GetId(pMsg->hWinSrc);
            switch (Id) {
            case GUI_ID_BUTTON0: //STATUS
                WM_DeleteWindow(hDlg);
                hDlg = GUI_CreateDialogBox(_aDialogStatus,
GUI_COUNTOF
                (_aDialogStatus), &_DialogStatus, WM_HBKWIN, 0,
0);
                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON0);
                WM_DisableWindow(hItem);

                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON1);
                WM_EnableWindow(hItem);
                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON2);
```

```
                WM_EnableWindow(hItem);


                break;

            case GUI_ID_BUTTON1: //TEMP
                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON1);
                WM_DisableWindow(hItem);
                //BUTTON_SetSkin(hItem, BUTTON_SKIN_FLEX);

                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON0);
                WM_EnableWindow(hItem);
                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON2);
                WM_EnableWindow(hItem);
                WM_DeleteWindow(hDlg);
                hDlg = GUI_CreateDialogBox(_aDialogTemp,
GUI_COUNTOF(_aDialogTemp), &_DialogTemp, 0, 0, 0);
                break;

            case GUI_ID_BUTTON2: //CONTROL
                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON2);
                WM_DisableWindow(hItem);
                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON0);
                WM_EnableWindow(hItem);
                hItem = WM_GetDialogItem(pMsg->hWin,
GUI_ID_BUTTON1);
                WM_EnableWindow(hItem);

                WM_DeleteWindow(hDlg);
                hDlg = GUI_CreateDialogBox(_aDialogControl,
GUI_COUNTOF(_aDialogControl), &_DialogControl, 0, 0, 0);

                break;
            }
        }
        break;
    default:
        WM_DefaultProc(pMsg);
    }
}
/*****************************End
Menu*******************************************/



void samiWindow(void) {
    WM_SetCreateFlags(WM_CF_MEMDEV);

    WM_SetCallback(WM_HBKWIN, &_mainWindow);//redrawn function

    BUTTON_SetDefaultSkin   (BUTTON_SKIN_FLEX);//emWin default
skin type
```

```
    GUI_CreateDialogBox(_aDialogMenu, GUI_COUNTOF(_aDialogMenu),
&_DialogMenu, 0, 0, 0);


    while (1) {
        _countTimer();//time counter function
        GUI_SetFont(&GUI_FontRounded33);//set font type
        GUI_DispStringAt(" ", 280, 440);//set a position of a
string
        GUI_DispFloat (C_Temp,5);//current temperature its return
float value
        GUI_DispString(" °C");

        GUI_SetFont(&GUI_FontDigit19);
        GUI_SetBkColor(0X0C0C0C0);

        GUI_SetColor(GUI_WHITE);
        GUI_DispStringAt(" ", 650, 440);
        GUI_DispDec(Hour, 2);
        GUI_DispString(":");
        GUI_DispDec(Min, 2);
        GUI_DispString(":");
        GUI_DispDec(Sec, 2);
        GUI_Delay(1000);
        GUI_DispCEOL();
    }

}

    // GUI_SetTextMode(GUI_TEXTMODE_REV);
```