Embedded systems

SMART HOME PROJECT

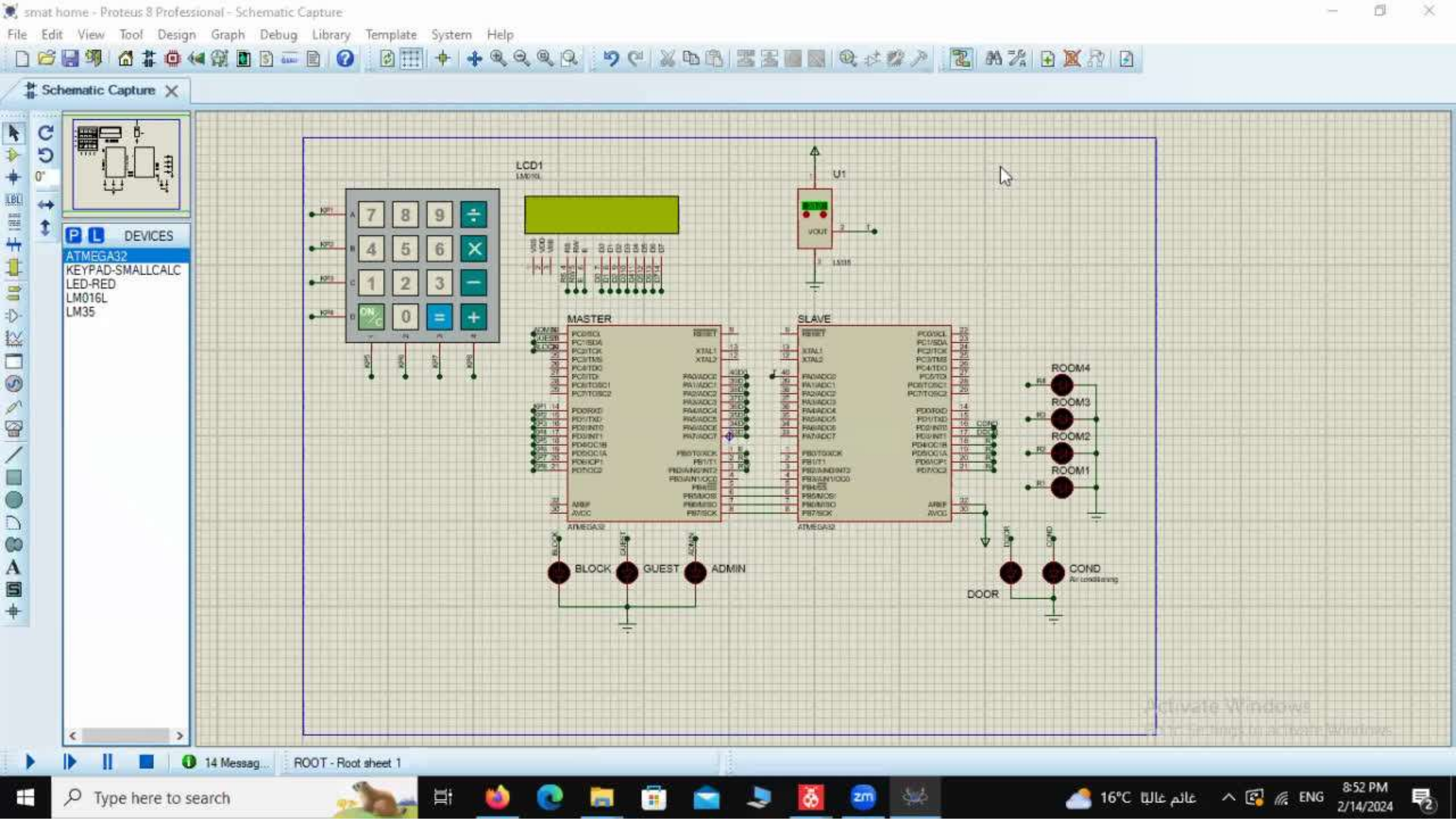
BY: Eng.Shahd Saied
Eng.Ahmed Rusrus

Table of CONTENTS

INTRODUCTION

A Smart Home is a residence equipped with devices and systems that automate tasks and provide remote control over household functions.

With Smart Home technology, you can control lighting, heating, security, and more manually or remotely.


smat home - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
DEVICES
ATMEGA32
KEYPAD-SMALLCALC
LED-RED
LM016L
LM35
LCD1
LM016L
U1
VOUT
LM35
MASTER
SLAVE
ATMEGA32
ATMEGA32
ROOM4
ROOM3
ROOM2
ROOM1
BLOCK
GUEST
ADMIN
DOOR
COND
Air conditioning
14 Messag...
ROOT - Root sheet 1
Activate Windows
Type here to search
16°C
ENG
8:52 PM
2/14/2024

MASTER CODE EXPLANATION

This code includes header files (#include) necessary for a microcontroller project. These headers likely provide functions and definitions for components like LCD, EEPROM, timers, keypad, SPI, LED, and menus. It sets up the groundwork for interfacing with these components in the main program logic.

```
#include "main_config.h"
#include "LCD.h"
#include "EEPROM.h"
#include "timer_driver.h"
#include "keypad_driver.h"
#include "SPI.h"
#include "LED.h"
#include "menu.h"
#include <avr/io.h>
```

MASTER CODE EXPLANATION

this part of code initializes variables for: Tracking session duration (session_counter) and timeout status (timeout_flag). Storing the average room temperature (temperature) and user-entered temperature digits (temp_ones and temp_tens). Managing login mode (login_mode) and whether login is blocked (block_mode_flag). Monitoring keypad input (key_pressed) to detect key presses.

```
volatile uint16 session_counter = 0;//indicate session time
uint8 timeout_flag = FALSE;//stores if the session is still valid or outdated

int main(void)
{
    uint8 temperature = 0;//The average temperature of the room
    uint8 temp_ones = NOT_SELECTED;//The entered right number of the temperature
    uint8 temp_tens = NOT_SELECTED;//The entered left number of the temperature

    uint8 login_mode = NO_MODE; //Store the current user mode admin or guest or not logged in
    uint8 block_mode_flag = FALSE;//is true if the login is blocked or false if not blocked
    uint8 key_pressed = NOT_PRESSED;//
    /*****************  INITIALIZE  **********************/
    LED_vInit(ADMIN_LED_PORT,ADMIN_LED_PIN);//initializes the led of admin
    LED_vInit(GUEST_LED_PORT,GUEST_LED_PIN);//initializes the led of guest
    LED_vInit(BLOCK_LED_PORT,BLOCK_LED_PIN);//initializes the led of block
    LCD_vInit();//initializes the LCD screen
    keypad_vInit();//initializes the keypad
    SPI_vInitMaster();//initializes the communication protocol of SPI
    /*****************************************************/
```

MASTER CODE EXPLANATION

Prints a welcome message on an LCD display: "Welcome to smart" on the first line and "home system" on the second line. Pauses program execution for 1000 milliseconds (1 second) using _delay_ms() function. Clears the LCD screen, removing all previously printed characters and resets the cursor to the beginning of the first row. Checks if admin and guest passwords are set in the EEPROM memory. If either or both passwords are not set, it displays a message prompting to set the admin password and informs that it's the first-time login.

```
/* Printing Welcome screen */
LCD_vSend_string("Welcome to smart");
LCD_movecursor(2,1);
LCD_vSend_string("home system");
_delay_ms(1000);//Halt the system for the given time in (ms)
LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the fi
/*************************/
/*Setting Admin and Guest passwords if not set */
//read the state of the the passwords of the admin and guest if both are set or not set
if ( (EEPROM_ui8ReadByteFromAddress(ADMIN_PASS_STATUS_ADDRESS)!=PASS_SET) || (EEPROM_ui8ReadByteFromAddress(GUEST_PASS_ST
{
    LCD_vSend_string("Login for");//printing login menu
    LCD_movecursor(2,1);//move the cursor to the second line
    LCD_vSend_string("first time");
    _delay_ms(1000);//Halt the system for the given time in (ms)
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of th
    LCD_vSend_string("Set Admin pass");//printing the set admin password menu
    LCD_movecursor(2,1);
    LCD_vSend_string("Admin pass:");
```

MASTER CODE EXPLANATION

Initializes variables to track the password input
process: pass_counter counts the characters of the password, and pass is an array that stores the password. Enters a loop to receive user input for the password until the required number of characters is entered. Within the loop, it waits for the user to press a key on the keypad. Once a key is pressed, it adds the pressed key to the pass array, displays the entered character on the LCD, and replaces it with a password symbol (e.g., asterisk '*'). Pauses program execution briefly to allow the user to preview the character entered as a password

```
/****************************** setting Admin password ****************************************/
uint8 pass_counter=0;//the counter of the characters of the password
uint8 pass[PASS_SIZE]={NOT_STORED,NOT_STORED,NOT_STORED,NOT_STORED};//the array where it stored the password
while (pass_counter<PASS_SIZE)//loop till the user finish inserting the pass
{
    key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial value
    while (key_pressed == NOT_PRESSED)//repeat till the user press any key
    {
        key_pressed = keypad_u8check_press();//if the user pressed any button in keypad save the value in key_pressed
    }

    pass[pass_counter]=key_pressed;//add the entered character to the pass array
    LCD_vSend_char(key_pressed);//print the entered character
    _delay_ms(CHARACTER_PREVIEW_TIME);//Halt the system for the given time in (ms)
    LCD_movecursor(2,12+pass_counter);//move the lcd cursor to the previous location to write the password symbol over
    LCD_vSend_char(PASSWORD_SYMBOL); // to display (Password sign *)
    _delay_ms(100);//Halt the system for the given time in (ms)
    pass_counter++;//increase the characters count
}
EEPROM_vWriteBlockToAddress(EEPROM_ADMIN_ADDRESS,pass,PASS_SIZE);//save the entire password as a block to the EEPROM
EEPROM_vWriteByteToAddress(ADMIN_PASS_STATUS_ADDRESS,PASS_SET);//write the status of pass as it is set
LCD clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
```

MASTER CODE EXPLANATION

Increments the pass_counter variable to track the number of characters entered for the password. Writes the entire password to the EEPROM memory and sets the status of the admin password as "set". Clears the LCD screen and displays a "Pass Saved“ message briefly. Resets the pass_counter for setting the guest password, then prompts the user to enter the guest password similarly to how the admin password was set..

```
LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the

/******************************** setting guest password ******************************************/
pass_counter=0;//reset password counter which count the characters of the pass
LCD_vSend_string("Set Guest Pass");//printing the set admin password menu
LCD_movecursor(2,1);
LCD_vSend_string("Guest Pass:");
while (pass_counter<PASS_SIZE)//loop till the user finish inserting the pass
{
    key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial value
    while (key_pressed == NOT_PRESSED)//repeat till the user press any key
    {
        key_pressed = keypad_u8check_press();//if the user pressed any button in keypad save the value in key_pressed
    }

    pass[pass_counter]=key_pressed;//add the entered character to the pass array
    LCD_vSend_char(key_pressed);//print the entered character
    _delay_ms(CHARACTER_PREVIEW_TIME);//Halt the system for the given time in (ms)
    LCD_movecursor(2,12+pass_counter);//move the lcd cursor to the previous location to write the password symbol over
    LCD_vSend_char(PASSWORD_SYMBOL); // to display (Password sign *)
    _delay_ms(100);//Halt the system for the given time in (ms)
    pass_counter++;//increase the characters count
}
```

MASTER CODE EXPLANATION

Writes the entire password block to an EEPROM address, followed by updating the status of the password setting, and displays a "Pass Saved" message on an LCD screen. Then it resets a password counter and prompts the user to set a guest password on the LCD screen. Continuously reads keypad inputs, appending them to the password array until it reaches the specified password size, simultaneously displaying each entered character on the LCD screen.

```
EEPROM_vWriteBlockToAddress(EEPROM_ADMIN_ADDRESS,pass,PASS_SIZE);//save the entire password as a block to the EEPROM
EEPROM_vWriteByteToAddress(ADMIN_PASS_STATUS_ADDRESS,PASS_SET);//write the status of pass as it is set
LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
LCD_vSend_string("Pass Saved");// show pass saved message
_delay_ms(500);//Halt the system for the given time in (ms)
LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the

/****************************** setting guest password ****************************************/
pass_counter=0;//reset password counter which count the characters of the pass
LCD_vSend_string("Set Guest Pass");//printing the set admin password menu
LCD_movecursor(2,1);
LCD_vSend_string("Guest Pass:");
while (pass_counter<PASS_SIZE)//loop till the user finish inserting the pass
{
    key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial value
    while (key_pressed == NOT_PRESSED)//repeat till the user press any key
    {
        key_pressed = keypad_u8check_press();//if the user pressed any button in keypad save the value in key_pressed
    }

    pass[pass_counter]=key_pressed;//add the entered character to the pass array
    LCD_vSend_char(key_pressed);//print the entered character
```

MASTER CODE EXPLANATION

Writes the guest password to EEPROM and sets the status of the guest password as "set". Displays a "Pass Saved" message on the LCD, waits briefly, and then clears the screen. Resets the login block status in EEPROM and initializes a variable to count failed login attempts. Enters an infinite loop to handle periodic tasks, such as checking for key presses and managing login attempts and block status.

.

```
        LCD_vSend_char(key_pressed);//print the entered character
        _delay_ms(CHARACTER_PREVIEW_TIME);//Halt the system for the given time in (ms)
        LCD_movecursor(2,12+pass_counter);//move the lcd cursor to the previous location to write the password symbol over
        LCD_vSend_char(PASSWORD_SYMBOL); // to display (Password sign *)
        _delay_ms(100);//Halt the system for the given time in (ms)
        pass_counter++;//increase the characters count
    }
    EEPROM_vWriteBlockToAddress(EEPROM_GUEST_ADDRESS,pass,PASS_SIZE);//save the entire password as a block to the EEPROM
    EEPROM_vWriteByteToAddress(GUEST_PASS_STATUS_ADDRESS,PASS_SET);//write the status of pass as it is set
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
    LCD_vSend_string("Pass Saved");//move the lcd cursor to the previous location
    _delay_ms(500);//Halt the system for the given time in (ms)
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
    EEPROM_vWriteByteToAddress(LOGIN_BLOCKED_ADDRESS,FALSE);
}//The end of if admin and guest password is set
else//this code of else run only if the system is not running for the first time (ADMIN and GUEST passwords are set)
{
    block_mode_flag = EEPROM_ui8ReadByteFromAddress(LOGIN_BLOCKED_ADDRESS); //read the blocked location from EEPROM
}
while (1)//The start of the periodic code
{
    key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial value
    uint8 pass_tries_count=0;//stores how many times the user tried to log in to the system and failed
```

MASTER CODE EXPLANATION

Stopping the session timer, resetting session counter, and clearing timeout flag. Logging the user out, resetting keypad state, and turning off user LEDs. Displaying a "Session Timeout" message on the LCD and pausing execution briefly.

```
if ( timeout_flag==TRUE )//check for timeout
{//if timeout flag was raised
    timer0_stop();//stop the timer that increase the session counter
    session_counter = 0;//clear session counter
    timeout_flag=FALSE;//clear time out flag
    login_mode=NO_MODE;//log the user out
    key_pressed = NOT_PRESSED;//clear the key_pressed to avoid unwanted selection in the menu switch
    LED_vTurnOff(GUEST_LED_PORT,GUEST_LED_PIN);//turnoff the led of the guest
    LED_vTurnOff(ADMIN_LED_PORT,ADMIN_LED_PIN);//turnoff the led of the admin
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of
    LCD_vSend_string("Session Timeout");//print session timeout message
    _delay_ms(1000);//Halt the system for the given time in (ms)
}
```

MASTER CODE EXPLANATION

Enters a loop where the user must select a login mode (admin or guest) before proceeding. Checks if the login process is blocked; if so, it displays a "Login blocked" message and waits for the block period to end. Prompts the user to select a login mode by displaying options on the LCD.

```
while (login_mode==NO_MODE)//The user can only leave the loop only in case of he was logged in as guest or admin
{
    if(block_mode_flag==TRUE)//if the login process was blocked wait till the end of the block period
    {
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column
        LCD_vSend_string("Login blocked");
        LCD_movecursor(2,1);
        LCD_vSend_string("wait 20 seconds");
        LED_vTurnOn(BLOCK_LED_PORT,BLOCK_LED_PIN);//Turn on the led of Blocked
        _delay_ms(BLOCK_MODE_TIME);//Halt the system for the given time in (ms)
        pass_tries_count = 0; //Clear the count on number of wrong tries
        block_mode_flag = FALSE;//Disable block of runtime
        LED_vTurnOff(BLOCK_LED_PORT,BLOCK_LED_PIN);//Turn off the led of Blocked
        EEPROM_vWriteByteToAddress(LOGIN_BLOCKED_ADDRESS,FALSE);//write false at blocked location in EEPROM
    }
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of
    LCD_vSend_string("Select mode :");
    LCD_movecursor(2,1);
    LCD_vSend_string("0:Admin 1:Guest");
    while(key_pressed==NOT_PRESSED)//wait for the selection of the mode
```

MASTER CODE EXPLANATION

This code waits for the user to press a key on the keypad before proceeding and then initializes variables for counting password characters and temporarily storing entered and stored passwords.

```
while(key_pressed==NOT_PRESSED)//wait for the selection of the mode
{
    key_pressed = keypad_u8check_press();//if the user pressed any button in keypad save the value in key_pressed
}
if ( key_pressed!=CHECK_ADMIN_MODE && key_pressed!=CHECK_GUEST_MODE )
{
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column
    LCD_vSend_string("Wrong input.");//Prints error message on the LCD
    key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial value
    _delay_ms(1000);//Halt the system for the given time in (ms)
    continue;//return to the loop of login #while (login_mode==NO_MODE)# line 128
}

uint8 pass_counter=0;//counts the entered key of the password from the keypad
uint8 pass[PASS_SIZE]={NOT_STORED,NOT_STORED,NOT_STORED,NOT_STORED};//temporarily hold the entire password that wil
uint8 stored_pass[PASS_SIZE]={NOT_STORED,NOT_STORED,NOT_STORED,NOT_STORED};//temporarily hold the entire stored pas
```

MASTER CODE EXPLANATION

Displaying "Admin mode" and prompting for password entry on the LCD. Waits for the user to enter a key on the keypad to input the password character by character. Displays the entered character on the LCD as a password symbol and pauses briefly.

.

```
switch(key_pressed)
{
    /****************************** Admin login ******************************************/
    case CHECK_ADMIN_MODE:
    while(login_mode!=ADMIN)//this loop is to repeat the login for admin in case of wrong password
    {
        key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial valu
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first c
        LCD_vSend_string("Admin mode");
        LCD_movecursor(2,1);
        LCD_vSend_string("Enter Pass:");
        _delay_ms(200);//Halt the system for the given time in (ms)
        pass_counter=0;//counts the number of entered characters
        while(pass_counter<PASS_SIZE)
        {
            while (key_pressed == NOT_PRESSED)//repeat till the user press any key
            {
                key_pressed = keypad_u8check_press();//if the user pressed any button in keypad save the value in
            }
            pass[pass_counter]=key_pressed;//add the entered character to the pass array
            LCD_vSend_char(key_pressed);//print the entered character
             delay ms(CHARACTER PREVIEW TIME);//Halt the system for the given time in (ms)
```

MASTER CODE EXPLANATION

Increments the password counter to track the number of characters entered. Resets the variable holding the pressed key to its initial value. Reads the stored admin password from EEPROM and compares it with the entered password. If correct, sets the login mode to admin, clears the LCD, displays a confirmation message, turns on the admin LED, starts the session timer, and clears the LCD again. If incorrect, increments the count of wrong login attempts and sets the login mode to "no mode".

```
        pass_counter++;//increase the password counter that count the characters of the pass
        key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial v
    }
    EEPROM_vReadBlockFromAddress(EEPROM_ADMIN_ADDRESS,stored_pass,PASS_SIZE);//read the stored pass from the EE

    /*compare passwords*/
    if ((ui8ComparePass(pass,stored_pass,PASS_SIZE)) == TRUE)//in case of right password
    {
        login_mode = ADMIN;//set the login mode to admin mode
        pass_tries_count=0;//clear the counter of wrong tries
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the firs
        LCD_vSend_string("Right pass");
        LCD_movecursor(2,1);
        LCD_vSend_string("Admin mode");
        _delay_ms(500);//Halt the system for the given time in (ms)
        LED_vTurnOn(ADMIN_LED_PORT,ADMIN_LED_PIN);//turn on the led of admin
        timer0_initializeCTC();//start the timer that counts the session time
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the firs
    }
    else//in case of wrong password
    {
        pass_tries_count++;//increase the number of wrong tries to block login if it exceeds the allowed tries
        login_mode = NO_MODE;//set the mode as not logged in
```

MASTER CODE EXPLANATION

Clears the LCD screen and displays the prompt "Select mode: 0:Admin 1:Guest" to the user. Waits for the user to select a mode by pressing a key on the keypad, pausing for 1 second between checks. Continues looping if no key is pressed, ensuring the user selects a mode before proceeding. Initializes variables to count password characters and store entered and stored passwords, depending on the mode selected by the user.

```
                LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the firs
                LCD_vSend_string("Wrong Pass");
                LCD_movecursor(2,1);
                LCD_vSend_string("Tries left:");
                LCD_vSend_char(TRIES_ALLOWED-pass_tries_count+ASCII_ZERO);//print the number of tries left before block
                _delay_ms(1000);//Halt the system for the given time in (ms)
                LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the firs
                if (pass_tries_count>=TRIES_ALLOWED)//if the condition of the block mode is true
                {
                    EEPROM_vWriteByteToAddress(LOGIN_BLOCKED_ADDRESS,TRUE);//write to the EEPROM TRUE to the the block
                    block_mode_flag = TRUE;//turn on block mode
                    break;//break the loop of admin login #while(login_mode!=ADMIN)# at line 169
                }
            }
        }
        break;//bREAK SWITCH case
        /****************************** Guest login ******************************************/
        case  CHECK_GUEST_MODE:
        while(login_mode != GUEST)
        {
            key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial value
            LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first co
            LCD_vSend_string("Guest mode");
```

MASTER CODE EXPLANATION

The LCD is moved to the second row, and "Enter Pass:" is displayed to prompt the user to enter a password. The system waits for the user to press keys on the keypad to input the password character by character, displaying each character as an asterisk (*) on the LCD. The entered password is compared with the stored admin password in EEPROM, and if it matches, the login mode is set to admin.

```
LCD_movecursor(2,1);
LCD_vSend_string("Enter pass:");
_delay_ms(200);//Halt the system for the given time in (ms)
pass_counter=0;//counts the number of entered characters
while(pass_counter<PASS_SIZE)
{
    while (key_pressed == NOT_PRESSED)//repeat till the user press any key
    {
        key_pressed = keypad_u8check_press();//if the user pressed any button in keypad save the value in key_pressed
    }
    pass[pass_counter]=key_pressed;//add the pressed key to the password string
    LCD_vSend_char(key_pressed);//print the entered character
    _delay_ms(CHARACTER_PREVIEW_TIME);//Halt the system for the given time in (ms)
    LCD_movecursor(2,12+pass_counter);//return the cursor to the location of the previous character to replace it with the
    LCD_vSend_char(PASSWORD_SYMBOL); // to display (Password sign *)
    _delay_ms(100);//Halt the system for the given time in (ms)
    pass_counter++;//increase the password counter that count the characters of the pass
    key_pressed = NOT_PRESSED;//return the variable that holds the pressed key from keypad to its initial value
}
EEPROM_vReadBlockFromAddress(EEPROM_GUEST_ADDRESS,stored_pass,PASS_SIZE);//Save the entire password in the EEPROM

/*compare passwords*/
if (ui8ComparePass(pass,stored_pass,PASS_SIZE)==TRUE)//in case of right password
```

MASTER CODE EXPLANATION

Sets the login mode to admin and resets the count of wrong login attempts. Clears the LCD screen and displays a "Right pass" message, followed by "Admin mode" to indicate successful login. Turns on the admin LED, starts the session timer, and then clears the LCD again. If the password is incorrect, increments the count of wrong login attempts, sets the login mode to "no mode," and displays a "Wrong Pass" message along with the number of tries left before blocking.

```
{
    login_mode = GUEST;
    pass_tries_count=0;//clear the counter of wrong tries
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
    LCD_vSend_string("Right pass");
    LCD_movecursor(2,1);
    LCD_vSend_string("Guest mode");
    _delay_ms(500);//Halt the system for the given time in (ms)
    LED_vTurnOn(GUEST_LED_PORT,GUEST_LED_PIN);//turn the led of gust mode that is connected to the master micro controller
    timer0_initializeCTC();//start the counter of the session
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
}
else//in case of wrong password
{
    pass_tries_count++;//increase the number of wrong tries to block login if it exceeds the allowed tries
    login_mode = NO_MODE;//set the mode as not logged in
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
    LCD_vSend_string("Wrong pass");
    LCD_movecursor(2,1);
    LCD_vSend_string("Tries left:");
    LCD_vSend_char(TRIES_ALLOWED-pass_tries_count+ASCII_ZERO);//print the number of tries left before block mode to be act
    _delay_ms(1000);//Halt the system for the given time in (ms)
    LCD clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
```

MASTER CODE EXPLANATION

Displays a "Wrong pass" message on the LCD along with the number of tries left before block mode activation. Pauses briefly and then clears the LCD. Checks if the number of wrong login attempts exceeds the allowed threshold and activates block mode if true, breaking out of the loop for guest login.

```
        LCD_vSend_string("Wrong pass");
        LCD_movecursor(2,1);
        LCD_vSend_string("Tries left:");
        LCD_vSend_char(TRIES_ALLOWED-pass_tries_count+ASCII_ZERO);//print the number of tries left before block mode to be act
        _delay_ms(1000);//Halt the system for the given time in (ms)
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
        if (pass_tries_count>=TRIES_ALLOWED)//if the condition of the block mode is true
        {
            EEPROM_vWriteByteToAddress(LOGIN_BLOCKED_ADDRESS,TRUE);//write to the EEPROM TRUE to the the block mode address
            block_mode_flag = TRUE;//turn on block mode
            break;//breaks the loop of insert guest password #while(login_mode != GUEST)# line 228
        }//end of if that check if the number of tries exceeds the maximum tries allowed
    }//end of the case of wrong password
}//end of loop of guest login
break;//break for CHECK_GUEST_MODE switch case
nd of switch



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

ow_menu = MAIN_MENU;
```

MASTER CODE EXPLANATION

Sets a variable show_menu to indicate which menu to display (initially set to MAIN_MENU). Enters a loop to continuously display the menu as long as the session timeout flag is not raised. Clears the LCD and displays the main menu options depending on the login mode (admin or guest), including options for different rooms.

```
ow_menu = MAIN_MENU;



meout_flag!=TRUE)//Show the menu in case of the time is not out

pressed = NOT_PRESSED;//Set the key pressed by the user to its default value
ch (show_menu)

case MAIN_MENU:
do
{
    /****************** print main Menu *****************/
    LCD_clearscreen();
    LCD_vSend_string("1:Room1 2:Room2");
    LCD_movecursor(2,1);
    if(login_mode==ADMIN)//check login mode
    {
        LCD_vSend_string("3:Room3 4:More ");//this menu options only printed if the logged in user is an admin
    }
    else if(login_mode==GUEST)//check login mode
    {
        LCD_vSend_string("3:Room3 4:Room4");//this menu options only printed if the logged in user is a guest
    }
```

MASTER CODE EXPLANATION

Waits for a key press and then determines the next menu based on the pressed key and user's login mode. Includes a brief delay to prevent duplicate key presses. Assigns the next menu to be displayed according to the pressed key and user's privileges.

```
key_pressed = u8GetKeyPressed(login_mode);//wait for the user till key is pressed or the time is out
_delay_ms(100);//to avoid the duplication of the pressed key

if (key_pressed == SELECT_ROOM1)//If key pressed is 1
{
    show_menu = ROOM1_MENU;//Set the next menu to be shown to room1 menu
}
else if (key_pressed == SELECT_ROOM2)//If key pressed is 2
{
    show_menu = ROOM2_MENU;//Set the next menu to be shown to room2 menu
}
else if (key_pressed == SELECT_ROOM3)//If key pressed is 3
{
    show_menu = ROOM3_MENU;//Set the next menu to be shown to room3 menu
}
else if (key_pressed == SELECT_ROOM4 && login_mode == GUEST)//If key pressed is 4 and the logged in user is guest
{
    show_menu = ROOM4_MENU;//Set the next menu to be shown to room4 menu
}
else if (key_pressed == ADMIN_MORE_OPTION && login_mode == ADMIN)//If key pressed is 4 and the logged in user is admin
{
    show_menu = MORE_MENU;//Set the next menu to be shown to more menu
}
```

MASTER CODE EXPLANATION

This part of the code handles displaying a wrong input message if the user presses an invalid key. It clears the LCD screen, prints the "Wrong input" message, and then delays execution for 500 milliseconds to give the user time to see the message. The code continues to loop until the user either enters a valid key (1 to 4) or until a timeout occurs, depending on the condition specified in the do-while loop.

```
    }
    else if(key_pressed != NOT_PRESSED)//show wrong input message if the user pressed wrong key
    {
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
        LCD_vSend_string("Wrong input");//print error message
        _delay_ms(500);//Halt the system for the given time in (ms)
    }
} while ( ((key_pressed < '1') || (key_pressed > '4') ) && (timeout_flag == FALSE) );//break the loop in case of valid key or

break;//End of main menu case

case MORE_MENU:
do
{
    /****************** print more Menu *****************/
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the fir
    LCD_vSend_string("1:Room4    2:DOOR   ");
    LCD_movecursor(2,1);
    LCD_vSend_string("3:Air Cond.4:RET");
    /****************************************************/
    key_pressed = u8GetKeyPressed(login_mode);//wait for the user till key is pressed or the time is out
    _delay_ms(100);//to avoid the duplication of the pressed key
```

MASTER CODE EXPLANATION

This code segment handles key presses and sets the next menu accordingly based on the pressed key or displays a "Wrong input" message if an invalid key is pressed, followed by a brief delay to allow the user to notice the message

```
if (key_pressed == SELECT_ROOM4_ADMIN)//If key pressed is 1
{
    show_menu = ROOM4_MENU;//Set the next menu to be shown to room4 menu
}
else if (key_pressed == SELECT_DOOR)//If key pressed is 2
{
    show_menu = DOOR_MENU;//Set the next menu to be shown to DOOR menu
}
else if (key_pressed == SELECT_AIR_CONDITIONING)//If key pressed is 3
{
    show_menu = AIRCONDITIONING_MENU;//Set the next menu to be shown to Air conditioning menu
}
else if (key_pressed == ADMIN_RET_OPTION)//If key pressed is 4 (RET)
{
    show_menu = MAIN_MENU;//Set the next menu to be shown to main menu
}
else if(key_pressed != NOT_PRESSED)//show wrong input message if the user pressed wrong key
{
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
    LCD_vSend_string("Wrong input");//print error message
    _delay_ms(500);//Halt the system for the given time in (ms)
}
```

MASTER CODE EXPLANATION

This part ensures the program waits until the user presses a valid key (1 to 4) or until a timeout occurs, then moves to the next menu. In the Air Conditioning menu case, it displays options for setting temperature and controlling the air conditioning, and waits for a key press to determine the next menu..

```
} while (( (key_pressed < '1') || (key_pressed > '4') ) && (timeout_flag == FALSE));//break the loop in case of valid key or

break;//End of more menu case

case AIRCONDITIONING_MENU:
do
{
    /****************** print more Menu *****************/
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the fi
    LCD_vSend_string("1:Set temperature ");
    LCD_movecursor(2,1);
    LCD_vSend_string("2:Control  0:RET");
    /****************************************************/
    key_pressed = u8GetKeyPressed(login_mode);//wait for the user till key is pressed or the time is out
    _delay_ms(100);//to avoid the duplication of the pressed key

    if (key_pressed == SELECT_SET_TEMPERATURE)//If key pressed is 1
    {
        show_menu = TEMPERATURE_MENU;//Set the next menu to be shown to set temperature menu
    }
    else if (key_pressed == SELECT_AIR_COND_CTRL)//If key pressed is 2
    {
        show menu = AIRCOND CTRL MENU;//Set the next menu to be shown to air conditioning control menu
```

MASTER CODE EXPLANATION

the Air Conditioning menu: If the user presses 0, it goes back to the More menu. If any other key is pressed, it shows a "Wrong input" message. It continues this loop until a valid key (0 to 2) is pressed or until a timeout occurs. In the Room 1 and Room 2 menu cases: It displays the respective room menu based on the user's login mode. After displaying the menu, it returns to the Main menu.

```
        show_menu = AIRCOND_CTRL_MENU;//Set the next menu to be shown to air conditioning control menu
    }
    else if (key_pressed == SELECT_AIR_COND_RET)//If key pressed is 0
    {
        show_menu = MORE_MENU;//Set the next menu to be shown to more menu
    }
    else if(key_pressed != NOT_PRESSED)//show wrong input message if the user pressed wrong key
    {
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the
        LCD_vSend_string("Wrong input");//print error message
        _delay_ms(500);//Halt the system for the given time in (ms)
    }
} while (( (key_pressed < '0') || (key_pressed > '2') ) && (timeout_flag == FALSE));//break the loop in case of valid key or t
break;//End of air conditioning menu case

case ROOM1_MENU:
vMenuOption(ROOM1_MENU,login_mode);//call the function that show the menu of room 1
show_menu = MAIN_MENU;//Set the next menu to be shown to main menu
break;//End of room1 menu case

case ROOM2_MENU:
vMenuOption(ROOM2_MENU,login_mode);//call the function that show the menu of room 2
show menu = MAIN MENU;//Set the next menu to be shown to main menu
```

MASTER CODE EXPLANATION

displays the menu options for Room 3 based on the user's login mode. After displaying the menu, it returns to the Main menu. In the Room 4 menu case: It displays the menu options for Room 4 based on the user's login mode. If a guest is logged in, it returns to the Main menu. If an admin is logged in, it goes to the More menu.

the Door menu case: It shows the Door menu options. It then moves to the More menu.

```
break;//End of room2 menu case

case ROOM3_MENU:
vMenuOption(ROOM3_MENU,login_mode);//call the function that show the menu of room 3
show_menu = MAIN_MENU;//Set the next menu to be shown to main menu
break;//End of room3 menu case

case ROOM4_MENU:
vMenuOption(ROOM4_MENU,login_mode);//call the function that show the menu of room 4
if (login_mode == GUEST)//in case of guest is logged in
{
    show_menu = MAIN_MENU;//Set the next menu to be shown to main menu
}
else//in case of admin is logged in
{
    show_menu = MORE_MENU;//Set the next menu to be shown to more menu
}
break;//End of room4 menu case

case DOOR_MENU:
vMenuOption(DOOR_MENU,login_mode);//call the function that show the menu of tv
show_menu = MORE_MENU;//Set the next menu to be shown to more menu
break;//End of TV menu case
```

MASTER CODE EXPLANATION

the Air Conditioning Control menu case: It displays the Air Conditioning Control menu options. After that, it switches to the Air Conditioning menu. In the Temperature menu case: It prompts the user to set the temperature. If the timeout occurs before the user sets the temperature, it moves to the next menu.

```
case AIRCOND_CTRL_MENU:
vMenuOption(AIRCOND_CTRL_MENU,login_mode);//call the function that show the menu of Air conditioning control
show_menu = AIRCONDITIONING_MENU;//Set the next menu to be shown to air conditioning menu
break;//End of air conditioning control menu case

case TEMPERATURE_MENU:
temperature = 0;//clear the value of temperature
while (temperature==0 && timeout_flag == FALSE)//start the loop that asks for the temperature from the user in case of the ti
{
    key_pressed = NOT_PRESSED;//set the key pressed to the default value
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column of the fi
    LCD_vSend_string("Set temp.:__ ");//print the format of inserting temperature
    LCD_vSend_char(DEGREES_SYMBOL); // print the symbol of degree
    LCD_vSend_char('C'); // print the C character
    LCD_movecursor(1,11);//move the cursor to the place to write the entered temperature
    _delay_ms(200);//Halt the system for the given time in (ms)
    /*****************************************************************************/
    key_pressed = u8GetKeyPressed(login_mode);//wait for the user till key is pressed or the time is out
    _delay_ms(250);//to avoid the duplication of the pressed key

    if (timeout_flag == TRUE) //in case of the time is out before the user press a key
    {
        break;//break the loop that ask for temperature
```

MASTER CODE EXPLANATION

the user enters a non-numeric value, it displays a "Wrong input" message on the LCD and halts briefly before continuing the loop. If the entered value is valid (numeric), it displays the value on the LCD and saves it. After handling the input, it waits for the user to press another key. If a timeout occurs, it breaks the loop asking for temperature input

```
if (key_pressed <'0' || key_pressed >'9')//show wrong input message if the user entered non numeric value
{
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column
    LCD_vSend_string("Wrong input");//print error message
    _delay_ms(500);//Halt the system for the given time in (ms)
    continue;//return to #while (temperature==0)# line 672
}
else//if the value is valid
{
    LCD_vSend_char(key_pressed);//print the value on the lcd
    temp_tens = key_pressed-ASCII_ZERO;//save the entered value
    key_pressed = NOT_PRESSED;//set the key pressed to the default value
}
/*****************************************************************************/
key_pressed = u8GetKeyPressed(login_mode);//wait for the user till key is pressed or the time is out
_delay_ms(250);//to avoid the duplication of the pressed key

if (timeout_flag == TRUE)//if the user session is timeout
{
    break;//break the loop that ask for temperature
}
if ((key_pressed <'0' || key_pressed >'9'))//show wrong input message if the user entered non numeric value
```

MASTER CODE EXPLANATION

the user inputs an invalid temperature value, it displays an error message and prompts for input again. For valid input, it sends the temperature setting via SPI and displays a confirmation message. After successful input or timeout, it returns to the Air Conditioning menu.

```
        LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first co
        LCD_vSend_string("Wrong input");//print error message
        _delay_ms(500);//Halt the system for the given time in (ms)
        continue;//repeat the loop that ask for the temperature
    }
    else//if the value is valid
    {
        LCD_vSend_char(key_pressed);//print the value on the lcd
        temp_ones = key_pressed-ASCII_ZERO;//save the entered value
        key_pressed = NOT_PRESSED;//set the key pressed to the default value
    }
    temperature = temp_tens*10 + temp_ones;//set the value of the temperature from the given separated values
    SPI_ui8TransmitRecive(SET_TEMPERATURE);//Send the code of set temperature
    _delay_ms(200);//Halt the system to prevent write collision
    SPI_ui8TransmitRecive(temperature);//send the entered temperature
    LCD_clearscreen();//remove all previously printed characters on the LCD and move the cursor to the first column
    LCD_vSend_string("Temperature Sent");//show the message
    _delay_ms(500);//Halt the system for the given time in (ms)
}
show_menu = AIRCONDITIONING_MENU;//Set the next menu to be shown to air conditioning menu
break;//break from switch
nd of the switch
f while that repeats the menu after each successful action till session timeout
```

SLAVE CODE
Explanation

This code includes various header files for communication protocols (SPI), standard message definitions, digital I/O operations, LED control, ADC functionality, timer configurations, and application-specific macros.

```
#include "SPI.h"
#include "STD_MESSAGES.h"
#include "DIO.h"
#include "LED.h"
#include "ADC_driver.h"
#include "timer_driver.h"
#include "APP_slave_Macros.h"
```

SLAVE CODE
Explanation

This code defines variables for storing the required temperature, current room temperature reading, a counter for determining ISR periodicity, and the last state of the air conditioning unit to facilitate hysteresis control.

```
volatile uint16 required_temperature=24; // the required temperature which sent from Master with initial value 24
volatile uint16 temp_sensor_reading=0; // the temperature of the room
volatile uint8 counter=0; // the counter which determine the periodic time of implementing ISR
volatile uint8 last_air_conditioning_value=AIR_CONDTIONING_OFF; // last air conditioning value which will help in
```

SLAVE CODE
Explanation

This main function initializes the ADC, Timer 0 in CTC mode, and SPI as a slave. It also initializes the output pins for LEDs or devices connected to represent various functionalities. It defines variables to store received data from the master and the response to be sent back.

```
int main(void)
{
    ADC_vinit();//initialize the ADC of the micro controller
    timer0_initializeCTC();//Initialize the timer zero of the micro controller
    SPI_vInitSlave();//initialize the SPI as a slave

    /* initialization of output pins of connected leds or devices */
    LED_vInit(AIR_COND_PORT,AIR_COND_PIN);
    LED_vInit(DOOR_PORT,DOOR_PIN);
    LED_vInit(ROOM1_PORT,ROOM1_PIN);
    LED_vInit(ROOM2_PORT,ROOM2_PIN);
    LED_vInit(ROOM3_PORT,ROOM3_PIN);
    LED_vInit(ROOM4_PORT,ROOM4_PIN);

    uint8 request = DEFAULT_ACK;//the value that is received from the master
    uint8 response = DEFAULT_ACK;//the values that is sent back to the master
```

SLAVE CODE
Explanation

In the main loop: It waits for a command from the master. If the command is to check the status of Room 1, it checks if the corresponding LED is on or off. It sends the status back to the master and continues waiting for further commands.

```
while(1)
{
    request = SPI_ui8TransmitRecive(DEFAULT_ACK);//wait for the master to start the transmitting
    //ALL Messages are defined in STD_messages.h
    switch (request)
    {
        /*******************************   STATUS COMMANDS ******************************/
        //commands related to send the current status back to the master
        case ROOM1_STATUS:
        if (LED_u8ReadStatus(ROOM1_PORT,ROOM1_PIN)==0)//if the led is turned off
        {
            response = OFF_STATUS;//set the response as off status
        }
        else if (LED_u8ReadStatus(ROOM1_PORT,ROOM1_PIN)==1)//if the led is turned on
        {
            response = ON_STATUS;//set the response as on status
        }
        else
        {
        }
        SPI_ui8TransmitRecive(response);//response to the transmitter with the status
        break;//break the switch case
        case ROOM2_STATUS:
```

SLAVE CODE
Explanation

For "ROOM2_STATUS", it checks the status of the LED for Room 2. If it's off, it responds with "OFF_STATUS"; if it's on, it responds with "ON_STATUS". It then sends the status back to the master and continues waiting for further commands. For "ROOM3_STATUS", it checks the status of the LED for Room 3. If it's off, it responds with "OFF_STATUS".

```
case ROOM2_STATUS:
if (LED_u8ReadStatus(ROOM2_PORT,ROOM2_PIN)==0)//if the led is turned off
{
    response = OFF_STATUS;//set the response as off status
}
else if (LED_u8ReadStatus(ROOM2_PORT,ROOM2_PIN)==1)//if the led is turned on
{
    response = ON_STATUS;//set the response as on status
}
else
{
}
SPI_ui8TransmitRecive(response);//response to the transmitter with the status
break;//break the switch case
case ROOM3_STATUS:
if (LED_u8ReadStatus(ROOM3_PORT,ROOM3_PIN)==0)//if the led is turned off
{
    response = OFF_STATUS;//set the response as off status
}
else if (LED_u8ReadStatus(ROOM3_PORT,ROOM3_PIN)==1)//if the led is turned on
{
    response = ON_STATUS;//set the response as on status
```

SLAVE CODE
Explanation

For "ROOM3_STATUS", it checks if the LED for Room 3 is on or off and responds accordingly. For

"ROOM4_STATUS", it checks if the LED for Room 4 is on or off and responds accordingly. For "AIR_COND_STATUS", it checks if the air conditioning unit is on or off and responds accordingly.

```
}
else
{
}
SPI_ui8TransmitRecive(response);//response to the transmitter with the status
break;//break the switch case
case ROOM4_STATUS:
if (LED_u8ReadStatus(ROOM4_PORT,ROOM4_PIN)==0)//if the led is turned off
{
    response = OFF_STATUS;//set the response as off status
}
else if (LED_u8ReadStatus(ROOM4_PORT,ROOM4_PIN)==1)//if the led is turned on
{
    response = ON_STATUS;//set the response as on status
}
else
{
}
SPI_ui8TransmitRecive(response);//response to the transmitter with the status
break;//break the switch case
case AIR_COND_STATUS:
if (LED_u8ReadStatus(AIR_COND_PORT,AIR_COND_PIN)==0)//if the led is turned off
```

SLAVE CODE
Explanation

For "AIR_COND_STATUS", it checks if the air conditioning unit is off, and if so, it responds with "OFF_STATUS". If it's on, it responds with "ON_STATUS". It then sends the status back to the master and continues waiting for further commands. For "TV_STATUS", it checks if the TV is off, and if so, it responds with "OFF_STATUS". If it's on, it responds with "ON_STATUS".

```
{
    response = OFF_STATUS;//set the response as off status
}
else if (LED_u8ReadStatus(AIR_COND_PORT,AIR_COND_PIN)==1)//if the led is turned on
{
    response = ON_STATUS;//set the response as on status
}
else
{
}
SPI_ui8TransmitRecive(response);//response to the transmitter with the status
break;
case DOOR_STATUS:
if (LED_u8ReadStatus(DOOR_PORT,DOOR_PIN)==0)//if the led is turned off
{
    response = OFF_STATUS;//set the response as off status
}
else if (LED_u8ReadStatus(DOOR_PORT,DOOR_PIN)==1)//if the led is turned on
{
    response = ON_STATUS;//set the response as on status
}
```

SLAVE CODE
Explanation

these cases: It transmits the response (status) back to the transmitter via SPI communication. It then breaks out of the switch case and continues with the loop. Regarding the "TURN ON COMMANDS": For each command, it turns on the respective LED or device. After performing the action, it breaks out of the switch case and continues with the loop.

```
else
{
}
SPI_ui8TransmitRecive(response);//response to the transmitter with the status
break;//break the switch case

/*******************************   TURN ON COMMANDS ******************************/
case ROOM1_TURN_ON:
LED_vTurnOn(ROOM1_PORT,ROOM1_PIN);//turn on the led of room 1
break;//break the switch case
case ROOM2_TURN_ON:
LED_vTurnOn(ROOM1_PORT,ROOM2_PIN);//turn on the led of room 2
break;//break the switch case
case ROOM3_TURN_ON:
LED_vTurnOn(ROOM3_PORT,ROOM3_PIN);//turn on the led of room 3
break;//break the switch case
case ROOM4_TURN_ON:
LED_vTurnOn(ROOM4_PORT,ROOM4_PIN);//turn on the led of room 4
break;//break the switch case
case AIR_COND_TURN_ON:
timer0_initializeCTC();
LED_vTurnOn(AIR_COND_PORT,AIR_COND_PIN);//turn on the led of air conditioning
break;//break the switch case
```

SLAVE CODE
Explanation

1.For the "DOOR_TURN_ON" command, it turns on the LED connected to the door and then continues with the loop. For the "TURN OFF COMMANDS": For each command, it turns off the respective LED or device. After performing the action, it continues with the loop.

```
case DOOR_TURN_ON:
LED_vTurnOn(DOOR_PORT,DOOR_PIN);//turn on the led of the TV
break;//break the switch case

/*******************************   TURN OFF COMMANDS *******************************/
case ROOM1_TURN_OFF:
LED_vTurnOff(ROOM1_PORT,ROOM1_PIN);//turn off the led of room 1
break;//break the switch case
case ROOM2_TURN_OFF:
LED_vTurnOff(ROOM2_PORT,ROOM2_PIN);//turn off the led of room 2
break;//break the switch case
case ROOM3_TURN_OFF:
LED_vTurnOff(ROOM3_PORT,ROOM3_PIN);//turn off the led of room 3
break;//break the switch case
case ROOM4_TURN_OFF:
LED_vTurnOff(ROOM4_PORT,ROOM4_PIN);//turn off the led of room 4
break;//break the switch case
case AIR_COND_TURN_OFF:
timer0_stop();
LED_vTurnOff(AIR_COND_PORT,AIR_COND_PIN);//turn off the led of air conditioning
break;//break the switch case
case DOOR_TURN_OFF:
LED_vTurnOff(DOOR_PORT,DOOR_PIN);//turn off the led of the TV
```

SLAVE CODE
Explanation

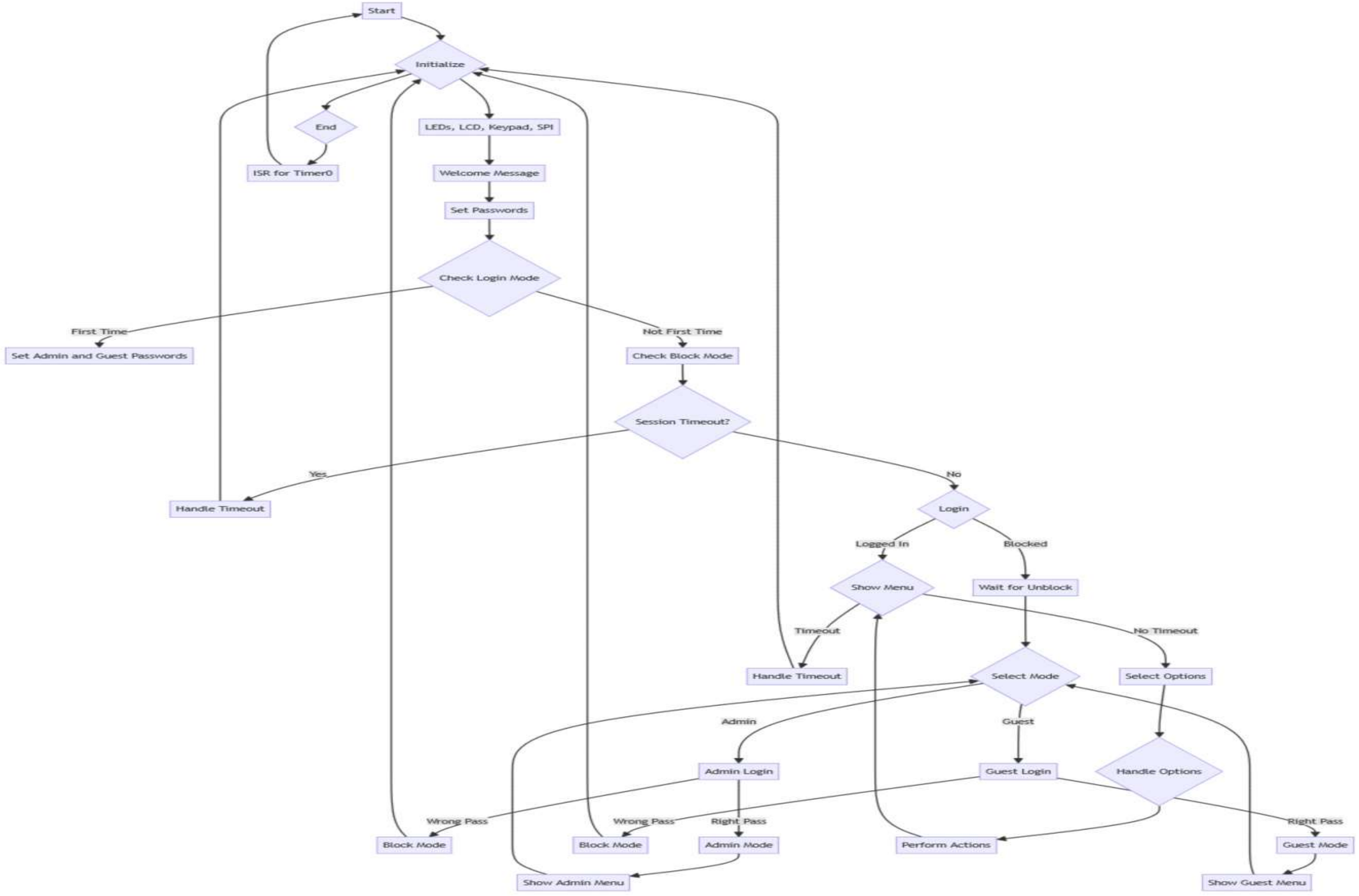
When receiving the "SET_TEMPERATURE" command, it gets the required temperature from the master and stores it. In the ISR: It counts the ticks of Timer 0. Every 10 ticks, it reads the temperature from a sensor. If the read temperature is one or more degrees higher than the required temperature, it executes subsequent code.

```
            break;//break the switch case
            case DOOR_TURN_OFF:
            LED_vTurnOff(DOOR_PORT,DOOR_PIN);//turn off the led of the TV
            break;//break the switch case

            /*******************************  Set temperature  *****************************/
            case SET_TEMPERATURE:
            required_temperature = SPI_ui8TransmitRecive(DEFAULT_ACK);//get the temperature from the master and store the temp
            break;//break the switch case
        }
    }
}

ISR(TIMER0_COMP_vect)
{
    counter++;//count the ticks of the timer zero
    if(counter>=10)//do that code every 10 ticks
    {
        counter=0;//clear the counter of ticks
        temp_sensor_reading=(0.25*ADC_u16Read());//read the temperature from the temperature sensor connected to the ADC of th
        if (temp_sensor_reading>=(required_temperature+1))//do that code if the read temperature if greater than required temp
        {
            LED_vTurnOn(AIR_COND_PORT,AIR_COND_PIN);//turn on the led of the air conditioning
```

SLAVE CODE
Explanation

the temperature is one or more degrees higher than the required temperature, it turns on the LED for the air conditioning and updates the last state value. If the temperature is one or more degrees lower than the required temperature, it turns off the LED for the air conditioning and updates the last state value. If the temperature matches the required temperature, it checks the last state value and either turns on or off the LED for the air conditioning accordingly.

```
if (temp_sensor_reading>=(required_temperature+1))//do that code if the read temperature if greater than required temperature
{
    LED_vTurnOn(AIR_COND_PORT,AIR_COND_PIN);//turn on the led of the air conditioning
    last_air_conditioning_value=AIR_CONDTIONING_ON;//save the value of the state of the air conditioning
}
else if (temp_sensor_reading<=(required_temperature-1))//do that code if the read temperature if lesser than required tempera
{
    LED_vTurnOff(AIR_COND_PORT,AIR_COND_PIN);//turn off the led of the air conditioning
    last_air_conditioning_value=AIR_CONDTIONING_OFF;//save the value of the state of the air conditioning
}
else if (required_temperature==temp_sensor_reading)//do that code if the read temperature is equal to the required temperatur
{
    if (last_air_conditioning_value==AIR_CONDTIONING_ON)//in the case of the last saved status of the air conditioning was on
    {
        LED_vTurnOn(AIR_COND_PORT,AIR_COND_PIN);//turn on the led of the air conditioning
    }
    else if (last_air_conditioning_value==AIR_CONDTIONING_OFF)//in the case of the last saved status of the air conditioning
    {
        LED_vTurnOff(AIR_COND_PORT,AIR_COND_PIN);//turn off the led of the air conditioning
    }
}
```


Start
Initialize
End
LEDs, LCD, Keypad, SPI
ISR for Timer0
Welcome Message
Set Passwords
Check Login Mode
First Time
Not First Time
Set Admin and Guest Passwords
Check Block Mode
Session Timeout?
Yes
No
Handle Timeout
Login
Logged In
Blocked
Show Menu
Wait for Unblock
Timeout
No Timeout
Handle Timeout
Select Mode
Select Options
Admin
Guest
Admin Login
Guest Login
Handle Options
Wrong Pass
Wrong Pass
Right Pass
Right Pass
Block Mode
Block Mode
Admin Mode
Perform Actions
Guest Mode
Show Admin Menu
Show Guest Menu

MASTER CODE FLOWCHART

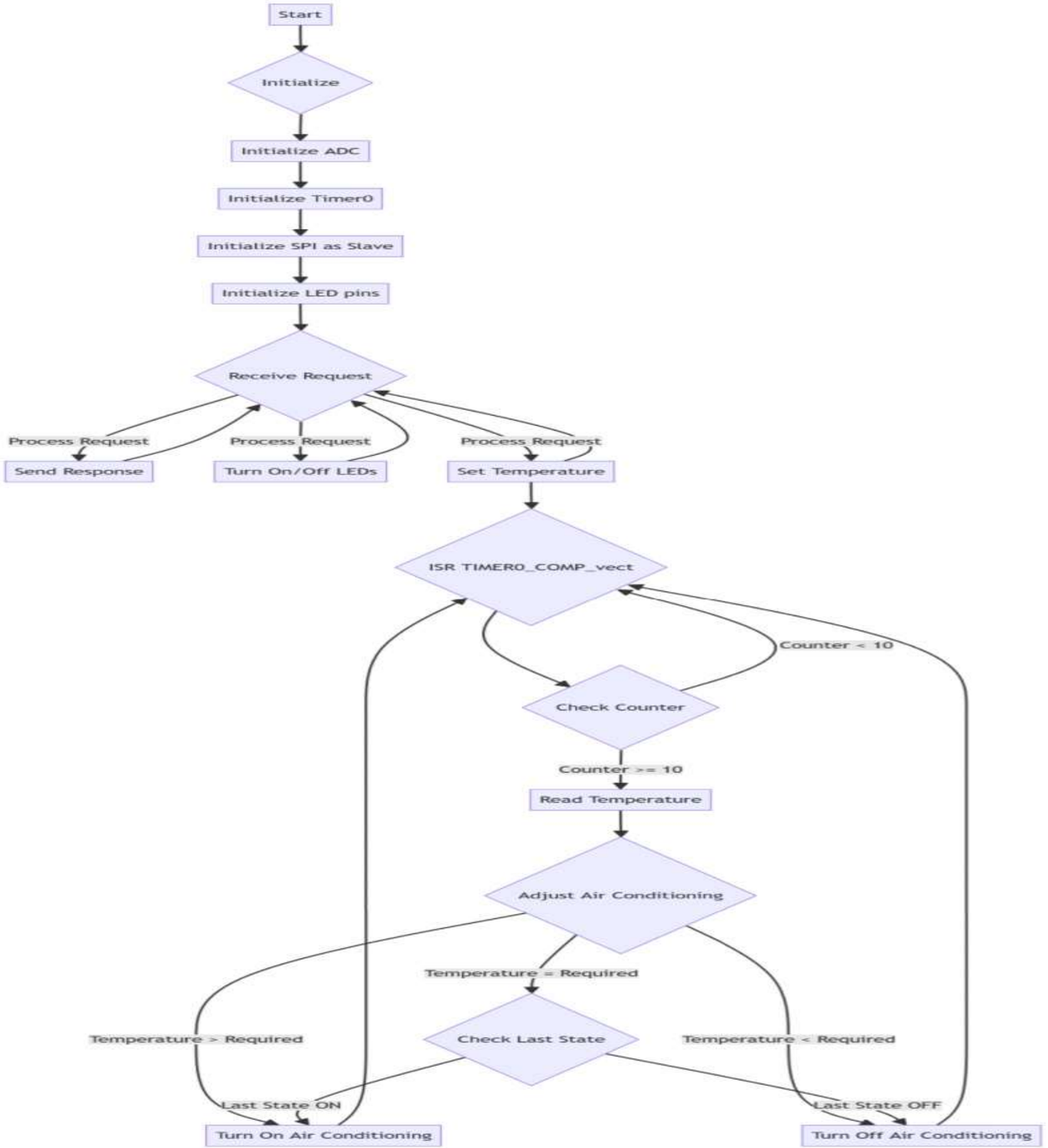
1. The code starts by including necessary header files and defining global variables.

2. In the `main` function:
- Initialization of LEDs, LCD, keypad, SPI, and other components is done.
- Welcome message is displayed on the LCD.
- If admin and guest passwords are not set, users are prompted to set them.
- If passwords are already set, the login mode is checked.

3. The main loop of the program starts with a while loop that runs indefinitely.
- It checks for session timeout and handles the timeout scenario.
- It then enters a loop for user login, where users can select admin or guest mode and enter passwords.
- After successful login, the user is directed to the main menu where different room options are displayed based on the user's mode.
- Users can navigate through different menus, set temperature, and control devices.

4. There are separate cases for different menus like main menu, more menu, air conditioning menu, room menus, etc.
- Users can select options within each menu based on the input provided.
- The program handles user input, displays messages, and performs actions accordingly.

5. There is an Interrupt Service Routine (ISR) for Timer0 Compare Match interrupt, which increments the session counter for session time tracking.


Start
Initialize
Initialize ADC
Initialize Timer0
Initialize SPI as Slave
Initialize LED pins
Receive Request
Process Request
Send Response
Process Request
Turn On/Off LEDs
Process Request
Set Temperature
ISR TIMER0_COMP_vect
Counter < 10
Check Counter
Counter >= 10
Read Temperature
Adjust Air Conditioning
Temperature = Required
Temperature > Required
Check Last State
Temperature < Required
Last State ON
Last State OFF
Turn On Air Conditioning
Turn Off Air Conditioning

The provided C program controls devices via SPI communication from a master device. Here's a condensed overview:

Includes necessary header files for SPI, standard messages, I/O, LED control, ADC, timer, and macros.

Declares global variables for temperature, sensor reading, ISR counter, and AC state.

.In the `main`: - Initializes ADC, timer, and SPI. - Initializes output pins. - Declares request and response variables. -

Waits for requests from the master via SPI in an infinite loop. - Executes actions based on received requests: status check

, turning devices on/off, and setting temperature.

SLAVE CODE FLOWCHART

Implements ISR for Timer0 Compare Match: - Increments counter on each call. - Reads temperature if counter reaches

10 - Controls AC LED based on temperature and required setting, with hysteresis to prevent rapid switching.

CONCLUSION

Smart Home technology offers a convenient, efficient, and secure way to manage your home. Embrace the future of living with Smart Home solutions! Thank you for joining us today.