Embedded Systems
(CCE407)

Secure Access Control System

using 8051 microcontroller

Report

Under the Supervision of:
Eng. Anas Elsayed

Presented By:

Name:	ID:
Farida Waheed Abdel Bary	221903168
Nourhan Farag Mohamed	231903707
Malak Mounier Abdellatif	231903643
Razan Ahmed Fawzy	221903165
Nour Hesham Elsayed	221902960
Lujain Ahmed Yousef	231903614
Raneem Ahmed Refaat	221903114

1. Project Overview:

This project presents a **dual-microcontroller home system** that demonstrates how different subsystems can operate independently while still maintaining seamless communication via a **serial communication protocol—specifically, I²C (Inter-Integrated Circuit)**.

The system is architected around two 8051-based microcontrollers, each responsible for a distinct function within the home environment:

- **Microcontroller 1** is dedicated to **external access control**, using a 4-digit keypad for password entry, an LCD for user interaction, and sensors for presence detection. It also includes security alerts via a buzzer and status indication through LEDs. Once the password is verified, it initiates communication with the second microcontroller.
- **Microcontroller 2** manages the **internal features** of the home, such as lighting modes and air conditioning. It uses push buttons for control and provides real-time updates on another LCD.

The key innovation of this project lies in the use of **I²C communication to synchronize these two microcontrollers**. I²C allows for simple yet robust **serial communication** using only two lines (SDA and SCL), making it ideal for embedded systems where pin efficiency and communication integrity are critical. In this project, a signal ('S') sent over I²C from Microcontroller 1 acts as a trigger to enable functionalities in Microcontroller 2.

2. I²C Communication in Our Home System:

2.1 Purpose of I²C in the Project

In our home system, we use **two separate microcontrollers**:

- **Microcontroller 1** handles **security and access** using a password.
- **Microcontroller 2** controls **AC and lighting** inside the home.

These two microcontrollers need a way to talk to each other. For this, we used a simple **I²C communication** setup. This allows Microcontroller 1 to tell Microcontroller 2 when someone has entered the correct password, so it can start the control functions.

2.2 How I²C is Set Up

- We used **two wires** for this connection:
  - **SDA (Data)** connected to **P0.6 on both MCUs.**
  - **SCL (Clock)** connected to **P0.7 on both MCUs.**
  - **Pull-up Resistors (10kΩ)** are connected to both SDA and SCL lines to maintain high idle levels.

We implemented the I²C protocol using software (also called "*bit-banging*"), because the 8051 microcontrollers don't have built-in I²C hardware.

2.3 I²C Master (Microcontroller 1)

After someone enters the correct password:

1. The LCD displays "*Open*" then "*Close*".
2. The microcontroller sends the character 'S' over I²C.
3. This 'S' tells Microcontroller 2 to activate mode.
4. The I²C functions used are:
   - *I2C_start ()* – Starts the transmission.
   - *I2C_write('S')* – Sends the character 'S'.
   - *I2C_stop ()* – Ends the transmission.

This is a **very simple use of I²C**, just to send a one-byte message.

2.4 I²C Slave (Microcontroller 2)

Microcontroller 2 keeps checking the I²C lines in a loop:

1. It watches SDA and SCL.
2. If **both go LOW**, it means Microcontroller 1 has started sending data.
3. It understands this as a trigger and:
    - Turns on the home screen on the LCD.
    - Enables the AC and lighting buttons.

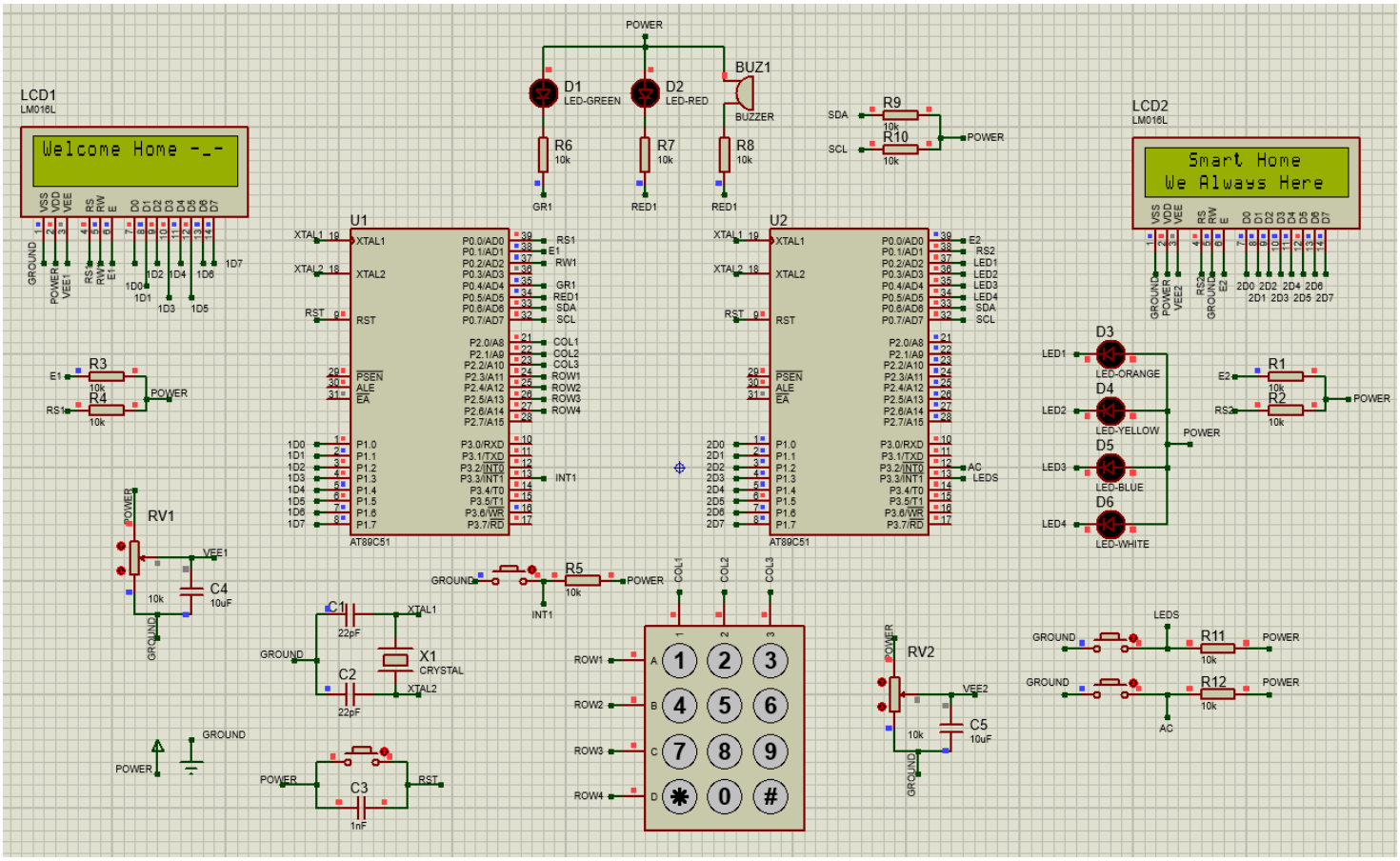
The function used here is:

- *I2C_Slave_Check ()* – Detects if data is coming and activates mode.

2.5 Why I²C is a Good Fit

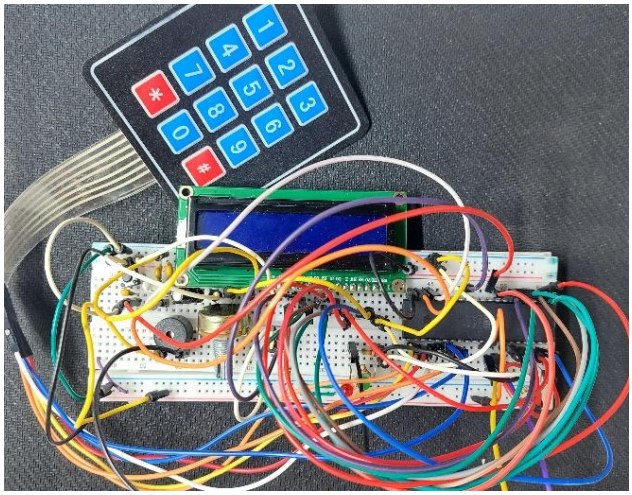
- **Only 2 wires** needed between the microcontrollers.
- Easy to implement in software.
- Allows for **simple, fast, and reliable communication**.
- Can be extended later to send more commands or data if needed.

3. Simulation Circuit Diagram:

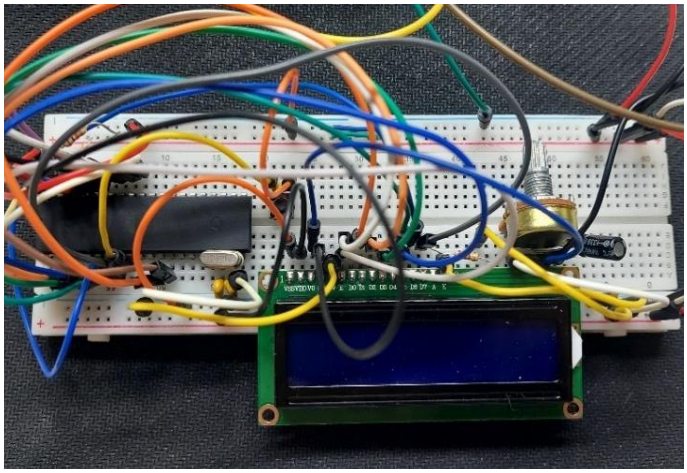


4. Circuit on real life:

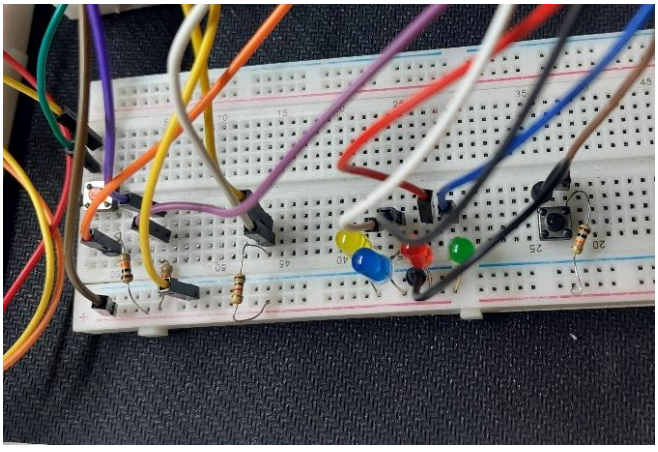
First Microcontroller Breadboard



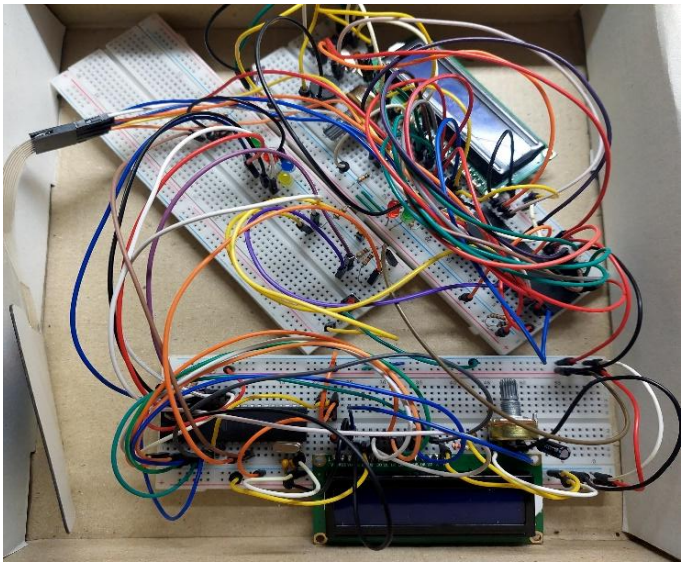
Second Microcontroller Breadboard



Serial Communication Breadboard



The Whole Project Connected



5. Algorithm:

- **Door Unit**

This unit handles password entry, verification, and access indication through LEDs and buzzer, and communication with the Inside Unit via I²C.

1. Initialize system peripherals: LCD, keypad, LEDs, and buzzer.
2. Display "***Welcome Home***" message on the LCD.
3. Wait for the user to enter a 4-digit password via the keypad.
4. Store the entered digits into memory.
5. Compare the entered password with the predefined correct password.
6. If the password is correct:
   - Turn ON the green LED. Display "***Open***" on the LCD. Wait for a short delay.
   - Display "***Close***" on the LCD.
   - **Send a message via I²C** to the Inside Home Unit to notify that access is granted.
   - This triggers the **screen change on the Inside Unit** from **"*Home - We're Always Here*"** to **"*AC State: / LEDs State:*"**.
7. If the password is incorrect:
   - Turn ON the red LED (and buzzer since connected in parallel).
   - Display "**Wrong Pass**" on the LCD.
   - Decrement the internal failed attempts counter.
   - If three consecutive incorrect attempts occur:
     - Display **"Intruder Alert"** on the LCD.
     - **Disable password entry** until reset.
     - Enter **alert mode**:
       - Start alternating **red and green LEDs** (blinking one ON while the other is OFF).
       - The **buzzer remains ON** (in parallel with red LED).
       - Continue this **LED blinking pattern** until the **reset button** is pressed.
     - Once the reset button is pressed:
       - Stop blinking.
       - Turn OFF both LEDs and buzzer.
       - Return to step 2 (await new input).

- **Inside Home Unit**

This unit handles control of room lighting and fan simulation through button inputs and responds to I²C messages from the Door Unit.

1. Initialize system components: LCD, buttons, and room simulation LEDs (White, Blue, Yellow, Orange).
2. Display **"*Home - We're Always Here*"** on the LCD.
3. Continuously check for **I²C message** from the Door Unit.
   If I²C message is received ('*S*'):
   - Change LCD display to **"*AC State: / LEDs State:*"**.
   - Enable user input for lighting and AC control.
4. Continuously monitor the button states:
   - If **Light button** is pressed:
     - Toggle White, Blue, Yellow LEDs.
     - Update LCD to show **"*LEDs State: M1*"** or **"*LEDs State: M2*"** or **"*LEDs State: M5*"** or **"*LEDs State: M4*"**.
   - If **AC button** is pressed:
     - Update LCD to show **"*AC State: ON*"** or **"*AC State: OFF*"**.
5. To reset:
   - A **reset button** to turn OFF devices and return to default display.
6.

6. Code Implementation:

➢ Microcontroller 1: Door Unit

This unit handles secure password entry, feedback via LEDs and buzzer, and communicates door status to the Inside Unit through I²C.

- **Main code**

```
#include <reg51.h>
#include "lcd.h"
#include "keypad.h"
#include "leds.h"
#include "delay.h"
#include "password.h"
#include "function.h"
#include "i2c.h"
#include "button.h"

unsigned char correct_pass_flag = 0;
unsigned char reEnter_pass_flag = 0;

void main() {
    unsigned char i;
    static bit welcome_shown = 0;

    LCD_init();
    I2C_init();
    LED_setRed(LED_OFF);
    LED_setGreen(LED_OFF);

    while (1) {
        if (Button_readStart() == BUTTON_PRESSED) {
            welcome_shown = 0;

            LCD_clear();
            LCD_writeText("Enter Password:");
            LCD_setCursor(1, 0);
            Password_inputFromUser();
            Password_checkCorrect();

            if (correct_pass_flag == 0) {
                LED_setGreen(LED_ON);
                LED_setRed(LED_OFF);
                num_ofCorrect_Password = 3;

                Password_successAction();
                I2C_start();
                I2C_write('S');
                I2C_stop();
                delay_ms(300);
            } else {
                num_ofCorrect_Password--;
                LED_setRed(LED_ON);
                LED_setGreen(LED_OFF);

                while (num_ofCorrect_Password != 0) {
                    LCD_clear();
                    LCD_writeText("WRONG PASS");
                    delay_ms(300);

                    LCD_clear();
                    LCD_writeText("Attempts Left:");
                    LCD_writeChar(num_ofCorrect_Password + '0');
                    delay_ms(300);

                    LCD_clear();
                    LCD_writeText("Enter Password:");
                    LCD_setCursor(1, 0);
                    Password_inputFromUser();
                    Password_checkCorrect();

                    if (correct_pass_flag == 1) {
                        num_ofCorrect_Password--;
                        if (num_ofCorrect_Password == 1) {
                            LED_setRed(LED_ON);
                        }
                        delay_ms(100);
                    } else if (correct_pass_flag == 0) {
                        reEnter_pass_flag = 1;
                        num_ofCorrect_Password = 3;
                        break;
                    }
                }
            }
```

```
            if (reEnter_pass_flag == 1) {
                reEnter_pass_flag = 0;
                LED_setGreen(LED_ON);
                LED_setRed(LED_OFF);

                Password_successAction();
                I2C_start();
                I2C_write('S');
                I2C_stop();
                delay_ms(300);
            }

            if (num_ofCorrect_Password == 0) {
                LCD_clear();
                LCD_writeText("ACCESS DENIED");
                LCD_setCursor(1, 0);
                LCD_writeText("INTRUDER ALERT!");
                delay_ms(1000);


                for (i = 0; i < 6; i++) {
                    if (i % 2 == 0) {
                        LED_setRed(LED_ON);
                        LED_setGreen(LED_OFF);
                    } else {
                        LED_setRed(LED_OFF);
                        LED_setGreen(LED_ON);
                    }
                    delay_ms(300);
                }

                while (1); // System locked; manual reset needed
            }
        } else if (!welcome_shown) {
            LCD_clear();
            LCD_writeText("Welcome Home -_-");
            welcome_shown = 1;
        }
    }
}
```

- **Function code**

```
#ifndef FUNCTION_H
#define FUNCTION_H

// Function Prototypes
void Password_inputFromUser(void);
void Password_checkCorrect(void);

#endif /* FUNCTION_H */
```

```
#include "function.h"
#include "lcd.h"
#include "keypad.h"
#include "delay.h"
#include "password.h"

extern unsigned char correct_pass_flag;

void Password_inputFromUser(void) {
    unsigned char i = 0, key;
    for (i = 0; i < 4; i++) entered_pass[i] = 0;

    i = 0;
    while (1) {
        key = Keypad_getKey();
        if (key == '#') break;

        if (key >= '0' && key <= '9' && i < 4) {
            entered_pass[i] = key;
            LCD_writeChar(entered_pass[i]);
            i++;
        } else if (key == '*' && i > 0) {
            i--;
            entered_pass[i] = 0;
            LCD_setCursor(1, i);
            LCD_writeChar(' ');
            LCD_setCursor(1, i);
        }

        delay_ms(50);
    }

    while (Keypad_getKey() != '#');
}

void Password_checkCorrect(void) {
    unsigned char i;
    correct_pass_flag = 0;
    for (i = 0; i < 4; i++) {
        if (entered_pass[i] != saved_pass[i])
            correct_pass_flag = 1;
    }
}
```

- **Password code**

```h
#ifndef PASSWORD_H
#define PASSWORD_H

// Global Variables (extern)
extern unsigned char saved_pass[5];
extern unsigned char entered_pass[5];
extern unsigned char num_ofCorrect_Password;

// Function Prototypes
void Password_successAction(void);

#endif /* PASSWORD H */
```

```c
#include "password.h"
#include "lcd.h"
#include "leds.h"
#include "delay.h"

// Define real memory for globals
unsigned char saved_pass[5] = {'1', '2', '3', '4', '\0'};
unsigned char entered_pass[5];
unsigned char num_ofCorrect_Password = 3;

void Password_successAction(void) {
    LCD_clear();
    LCD_writeText("Open");
    delay_ms(1000);

    LCD_clear();
    LCD_writeText("Close");
    delay_ms(500);

    LED_setGreen(LED_OFF);
    LCD_clear();
}
```

- **I²C code**

```h
#ifndef I2C_H
#define I2C_H

#include <reg51.h>

sbit SDA = P0^6;
sbit SCL = P0^7;


void I2C_init(void);
void I2C_start(void);
void I2C_stop(void);
void I2C_write(unsigned char byte);
bit I2C_WaitAck(void);


#endif
```

```c
#include "i2c.h"
#include "delay.h"

void I2C_init(void) {
    SDA = 1;
    SCL = 1;
}

void I2C_Start(void) {
    SDA = 1; SCL = 1;
    delay_ms(1);
    SDA = 0;
    delay_ms(1);
    SCL = 0;
}

void I2C_Stop(void) {
    SCL = 0; SDA = 0;
    delay_ms(1);
    SCL = 1;
    SDA = 1;
    delay_ms(1);
}

bit I2C_WaitAck(void) {
    SDA = 1;
    SCL = 1;
    delay_ms(1);
    if (SDA) {
        SCL = 0;
        return 0;
    }
    SCL = 0;
```

```
    return 1;
}
void I2C_Write(unsigned char dat) {
    unsigned char i;
    for (i = 0; i < 8; i++) {
        SDA = (dat & 0x80);
        SCL = 1;
        delay_ms(1);
        SCL = 0;
        dat <<= 1;
    }
    I2C_WaitAck();
}
```

- **Button code**

```
#ifndef BUTTON_H
#define BUTTON_H

#include <reg51.h>

// Define START button pin properly
sbit START_BUTTON_PIN = P3^3;

// Button states
#define BUTTON_PRESSED   1
#define BUTTON_RELEASED  0

// Function Prototype
unsigned char Button_readStart(void);

#endif
```

```
#include "button.h"
#include "delay.h"

unsigned char Button_readStart(void) {
    if (START_BUTTON_PIN == 0) {
        delay_ms(20);
        if (START_BUTTON_PIN == 0) {
            while (START_BUTTON_PIN == 0);
            return BUTTON_PRESSED;
        }
    }
    return BUTTON_RELEASED;
```

- **LEDs code**

```
#ifndef LEDS_H
#define LEDS_H

#include <reg51.h>

// LED Pin Definitions
sbit LED_RED = P0^5;
sbit LED_GREEN = P0^4;

// LED States
#define LED_ON   0
#define LED_OFF  1

// Function Prototypes
void LED_setRed(unsigned char state);
void LED_setGreen(unsigned char state);

#endif /* LEDS_H */
```

```
#include "leds.h"

void LED_setRed(unsigned char state) {
    LED_RED = state;
}

void LED_setGreen(unsigned char state) {
    LED_GREEN = state;
}
```

- **Delay code**

```
#ifndef DELAY_H
#define DELAY_H

void delay_ms(unsigned int ms);

#endif
```

```
#include "delay.h"
#include <reg51.h>

void delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 1275; j++);
}
```

- **LCD code**

```c
#ifndef LCD_H
#define LCD_H

#include <reg51.h>
#include "delay.h"

// LCD Control Pins using sbit
sbit LCD_RS = P0^0;
sbit LCD_E  = P0^1;
sbit LCD_RW = P0^2;

// LCD Data Port
#define LCD_DATA_PORT P1

// LCD Command Constants
#define LCD_CLEAR_COMMAND               0x01
#define LCD_GO_TO_HOME                  0x02
#define LCD_TWO_LINES_EIGHT_BITS_MODE   0x38
#define LCD_CURSOR_OFF                  0x0C
#define LCD_CURSOR_ON                   0x0E
#define LCD_SET_CURSOR_LOCATION         0x80

// Function Prototypes
void LCD_init(void);
void LCD_sendCommand(unsigned char command);
void LCD_writeChar(unsigned char character);
void LCD_writeText(const char *text);
void LCD_clear(void);
void LCD_setCursor(unsigned char row, unsigned char col);

#endif /* LCD_H */
```

```c
#include <reg51.h>
#include "delay.h"
#include "lcd.h"

void LCD_init(void) {
    LCD_RS = 0;
    LCD_E = 0;
    LCD_RW = 0;
    delay_ms(20);
    LCD_DATA_PORT = 0x00;

    LCD_sendCommand(LCD_TWO_LINES_EIGHT_BITS_MODE);
    LCD_sendCommand(LCD_CURSOR_OFF);
    LCD_sendCommand(LCD_CLEAR_COMMAND);
}

void LCD_sendCommand(unsigned char command) {
    LCD_RS = 0;     // Command mode
    LCD_RW = 0;     // Write mode
    delay_ms(1);
    LCD_E = 1;
    delay_ms(1);
    LCD_DATA_PORT = command;
    delay_ms(1);
    LCD_E = 0;
    delay_ms(1);
}

void LCD_writeChar(unsigned char character) {
    LCD_RS = 1;     // Data mode
    LCD_RW = 0;     // Write mode
    delay_ms(1);
    LCD_E = 1;
    delay_ms(1);
    LCD_DATA_PORT = character;
    delay_ms(1);
    LCD_E = 0;
    delay_ms(1);
}

void LCD_writeText(const char *text) {
    while (*text) {
        LCD_writeChar(*text++);
    }
}

void LCD_clear(void) {
    LCD_sendCommand(LCD_CLEAR_COMMAND);
}

void LCD_setCursor(unsigned char row, unsigned char col) {
    unsigned char memory_address;

    switch (row) {
        case 0: memory_address = col; break;
        case 1: memory_address = col + 0x40; break;
        default: memory_address = col; break;
    }
    LCD_sendCommand(memory_address | LCD_SET_CURSOR_LOCATION);
}
```

- **Keypad code**

```c
#ifndef KEYPAD_H
#define KEYPAD_H

#include <reg51.h>
#include "lcd.h"
#include "delay.h"

/* Define keypad configuration */
#define KEYPAD_NUM_COLS 3
#define KEYPAD_NUM_ROWS 4

/* Define ports for rows and columns */
#define KEYPAD_PORT P2

/* Button states */
#define KEYPAD_BUTTON_PRESSED  0
#define KEYPAD_BUTTON_RELEASED 1
```

```
/* Define pin mappings for rows */
sbit KEYPAD_ROW1 = KEYPAD_PORT^3;
sbit KEYPAD_ROW2 = KEYPAD_PORT^4;
sbit KEYPAD_ROW3 = KEYPAD_PORT^5;
sbit KEYPAD_ROW4 = KEYPAD_PORT^6;

/* Define pin mappings for columns */
sbit KEYPAD_COL1 = KEYPAD_PORT^0;
sbit KEYPAD_COL2 = KEYPAD_PORT^1;
sbit KEYPAD_COL3 = KEYPAD_PORT^2;

/* Function prototype */
unsigned char Keypad_getKey(void);

#endif /* KEYPAD_H */
```

```
#include "keypad.h"

/* Function to get the pressed key */
unsigned char Keypad_getKey(void) {
    while (1) {
        // Set all columns high
        KEYPAD_COL1 = 1;
        KEYPAD_COL2 = 1;
        KEYPAD_COL3 = 1;

        delay_ms(50);

        // Activate each row and scan columns
        KEYPAD_ROW1 = 0; KEYPAD_ROW2 = 1; KEYPAD_ROW3 = 1; KEYPAD_ROW4 = 1;
        if (KEYPAD_COL1 == 0) { delay_ms(50); while (KEYPAD_COL1 == 0); return '1'; }
        if (KEYPAD_COL2 == 0) { delay_ms(50); while (KEYPAD_COL2 == 0); return '2'; }
        if (KEYPAD_COL3 == 0) { delay_ms(50); while (KEYPAD_COL3 == 0); return '3'; }

        KEYPAD_ROW1 = 1; KEYPAD_ROW2 = 0; KEYPAD_ROW3 = 1; KEYPAD_ROW4 = 1;
        if (KEYPAD_COL1 == 0) { delay_ms(50); while (KEYPAD_COL1 == 0); return '4'; }
        if (KEYPAD_COL2 == 0) { delay_ms(50); while (KEYPAD_COL2 == 0); return '5'; }
        if (KEYPAD_COL3 == 0) { delay_ms(50); while (KEYPAD_COL3 == 0); return '6'; }

        KEYPAD_ROW1 = 1; KEYPAD_ROW2 = 1; KEYPAD_ROW3 = 0; KEYPAD_ROW4 = 1;
        if (KEYPAD_COL1 == 0) { delay_ms(50); while (KEYPAD_COL1 == 0); return '7'; }
        if (KEYPAD_COL2 == 0) { delay_ms(50); while (KEYPAD_COL2 == 0); return '8'; }
        if (KEYPAD_COL3 == 0) { delay_ms(50); while (KEYPAD_COL3 == 0); return '9'; }

        KEYPAD_ROW1 = 1; KEYPAD_ROW2 = 1; KEYPAD_ROW3 = 1; KEYPAD_ROW4 = 0;
        if (KEYPAD_COL1 == 0) { delay_ms(50); while (KEYPAD_COL1 == 0); return '*'; }
        if (KEYPAD_COL2 == 0) { delay_ms(50); while (KEYPAD_COL2 == 0); return '0'; }
        if (KEYPAD_COL3 == 0) { delay_ms(50); while (KEYPAD_COL3 == 0); return '#'; }
    }
}
```

➢ Microcontroller 2: Inside Home Unit

- **Main code**

```
#include <reg51.h>
#include "lcd.h"
#include "delay.h"
#include "button.h"
#include "leds.h"
#include "i2c_slave.h"

// Global variables
unsigned char AC_button_count = 0;
unsigned char Light_button_count = 0;
bit smart_mode = 0;

// Function prototypes
void LCD_showSmartFeatures(void);
void LCD_showHomeScreen(void);

void main(void) {
    LCD_init();
    I2C_Slave_Init();

    LED_setWhite(LED_OFF);
    LED_setBlue(LED_OFF);
    LED_setYellow(LED_OFF);
    LED_setOrange(LED_OFF);

    LCD_showHomeScreen();

    while (1) {
        I2C_Slave_Check();  // Check for I2C 'S'

        if (smart_mode) {
            // AC Button Handling
            if (Button_isACPressed() == BUTTON_PRESSED) {
                AC_button_count++;
                if (AC_button_count == 1) {
                    LCD_moveCursor(0, 10);
                    LCD_displayString("ON ");
                } else if (AC_button_count == 2) {
                    AC_button_count = 0;
                    LCD_moveCursor(0, 10);
                      LCD_displayString("OFF");
                  }
              }
```

```
            // Light Button Handling
            if (Button_isLightPressed() == BUTTON_PRESSE
                Light_button_count++;
                if (Light_button_count == 1) {
                    LCD_moveCursor(1, 14);
                    LCD_displayString("M1");
                    LED_setOrange(LED_ON);
                    LED_setYellow(LED_OFF);
                    LED_setBlue(LED_OFF);
                    LED_setWhite(LED_OFF);
                } else if (Light_button_count == 2) {
                    LCD_moveCursor(1, 14);
                    LCD_displayString("M2");
                    LED_setOrange(LED_ON);
                    LED_setYellow(LED_ON);
                    LED_setBlue(LED_OFF);
                    LED_setWhite(LED_OFF);
                } else if (Light_button_count == 3) {
                    LCD_moveCursor(1, 14);
                    LCD_displayString("M3");
                    LED_setOrange(LED_ON);
                    LED_setYellow(LED_ON);
                    LED_setBlue(LED_ON);
                    LED_setWhite(LED_OFF);
                } else if (Light_button_count == 4) {
                    LCD_moveCursor(1, 14);
                    LCD_displayString("M4");
                    LED_setOrange(LED_ON);
                    LED_setYellow(LED_ON);
                    LED_setBlue(LED_ON);
                    LED_setWhite(LED_ON);
                } else if (Light_button_count == 5) {
                    Light_button_count = 0;
                    LCD_moveCursor(1, 13);
                    LCD_displayString("OFF");
                    LED_setOrange(LED_OFF);
                   LED_setYellow(LED_OFF);
                   LED_setBlue(LED_OFF);
                   LED_setWhite(LED_OFF);
                }
            }
        }
    }
}

void LCD_showSmartFeatures(void) {
    LCD_clearScreen();
    LCD_moveCursor(0, 0);
    LCD_displayString("AC Status:   ");
    LCD_moveCursor(1, 0);
    LCD_displayString("Light Status:");
}

void LCD_showHomeScreen(void) {
    LCD_clearScreen();
    LCD_displayString("   Smart Home   ");
    LCD_moveCursor(1, 0);
    LCD_displayString(" We Always Here ");
}
```

- **LCD code**

```
#ifndef LCD_H
#define LCD_H

#include <reg51.h>
#include "delay.h"

// LCD Control Pins
sbit LCD_RS = P0^1;
sbit LCD_En = P0^0;

// LCD Data Port
#define LCD_DATA_PORT P1

// LCD Command Constants
#define LCD_CLEAR_COMMAND               0x01
#define LCD_GO_TO_HOME                  0x02
#define LCD_TWO_LINES_EIGHT_BITS_MODE   0x38
#define LCD_CURSOR_OFF                  0x0C
#define LCD_CURSOR_ON                   0x0E
#define LCD_SET_CURSOR_LOCATION         0x80

// LCD Function Prototypes
void LCD_init(void);
void LCD_sendCommand(unsigned char command);
void LCD_displayCharacter(unsigned char my_data);
void LCD_displayString(const char *Str);
void LCD_clearScreen(void);
void LCD_moveCursor(unsigned char row, unsigned char col);

#endif /* LCD_H */
```

```
#include <reg51.h>
#include "delay.h"
#include "lcd.h"

void LCD_init(void) {
    LCD_RS = 0;
    LCD_E = 0;
    LCD_RW = 0;
    delay_ms(20);
    LCD_DATA_PORT = 0x00;

```

```
    LCD_sendCommand(LCD_TWO_LINES_EIGHT_BITS_MODE);
    LCD_sendCommand(LCD_CURSOR_OFF);
    LCD_sendCommand(LCD_CLEAR_COMMAND);
}

void LCD_sendCommand(unsigned char command) {
    LCD_RS = 0;      // Command mode
    LCD_RW = 0;      // Write mode
    delay_ms(1);
    LCD_E = 1;
    delay_ms(1);
    LCD_DATA_PORT = command;
    delay_ms(1);
    LCD_E = 0;
    delay_ms(1);
}

void LCD_writeChar(unsigned char character) {
    LCD_RS = 1;      // Data mode
    LCD_RW = 0;      // Write mode
    delay_ms(1);
    LCD_E = 1;
    delay_ms(1);
    LCD_DATA_PORT = character;
    delay_ms(1);
    LCD_E = 0;
    delay_ms(1);
}

void LCD_writeText(const char *text) {
    while (*text) {
        LCD_writeChar(*text++);
    }
}

void LCD_clear(void) {
    LCD_sendCommand(LCD_CLEAR_COMMAND);
}

void LCD_setCursor(unsigned char row, unsigned char col) {
    unsigned char memory_address;

    switch (row) {
        case 0: memory_address = col; break;
        case 1: memory_address = col + 0x40; break;
        default: memory_address = col; break;
    }
    LCD_sendCommand(memory_address | LCD_SET_CURSOR_LOCATION);
}
```

- **LEDS code**

```
#ifndef LEDS_H
#define LEDS_H

#include <reg51.h>

// LED Pin Definitions
sbit LED_WHITE  = P0^5;
sbit LED_BLUE   = P0^4;
sbit LED_YELLOW = P0^3;
sbit LED_ORANGE = P0^2;

// LED States
#define LED_ON   0
#define LED_OFF  1

// Function Prototypes
void LED_setWhite(unsigned char status);
void LED_setBlue(unsigned char status);
void LED_setYellow(unsigned char status);
void LED_setOrange(unsigned char status);

#endif /* LEDS_H */
```

```
#include "leds.h"

void LED_setWhite(unsigned char status) {
    LED_WHITE = status;
}

void LED_setBlue(unsigned char status) {
    LED_BLUE = status;
}

void LED_setYellow(unsigned char status) {
    LED_YELLOW = status;
}

void LED_setOrange(unsigned char status) {
    LED_ORANGE = status;
}
```

- **BUTTON code**

```c
#include "button.h"
#include "delay.h"

unsigned char Button_isACPressed(void) {
    if (AC_button == BUTTON_PRESSED) {
        delay_ms(20);  // debounce delay
        if (AC_button == BUTTON_PRESSED) {
            while (AC_button == BUTTON_PRESSED); //
            return BUTTON_PRESSED;
        }
    }
    return BUTTON_RELEASED;
}

unsigned char Button_isLightPressed(void) {
    if (Light_button == BUTTON_PRESSED) {
        delay_ms(20);  // debounce delay
        if (Light_button == BUTTON_PRESSED) {
            while (Light_button == BUTTON_PRESSED);
            return BUTTON_PRESSED;
        }
    }
    return BUTTON_RELEASED;
}
```

```c
#ifndef BUTTON_H
#define BUTTON_H

#include <reg51.h>

// Define Button Pins
sbit AC_button = P3^2;
sbit Light_button = P3^3;

// Define Button States
#define BUTTON_PRESSED  0
#define BUTTON_RELEASED 1

// Function Prototypes
unsigned char Button_isACPressed(void);
unsigned char Button_isLightPressed(void);

#endif /* BUTTON_H */
```

- **I²C code**

```c
#ifndef I2C_SLAVE_H
#define I2C_SLAVE_H

void I2C_Slave_Init(void);
void I2C_Slave_Check(void);

#endif /* I2C_SLAVE_H */
```

```c
#include <reg51.h>
#include "i2c_slave.h"

sbit SDA = P0^6;
sbit SCL = P0^7;

extern bit smart_mode;
void LCD_showSmartFeatures(void);

void I2C_Slave_Init(void) {
    SDA = 1;
    SCL = 1;
}

void I2C_Slave_Check(void) {
    if (!SCL && !SDA) {
        smart_mode = 1;
        LCD_showSmartFeatures();
    }
}
```