BERFIN’S SMART HOME PROJECT

Buse Berfin Halefoğlu — 210401020

CENG313/Microprocessors
Instructor: Dr. Gözde Çay
Date: 19.05.2025

I. Introduction

This project aims to design and implement a basic smart home system using the AT89C51/AT89C52 microcontroller family. The system combines sensor data acquisition, access control, and real-time environmental monitoring to improve home security and energy efficiency.

The hardware includes various components such as an LM35 temperature sensor for environmental data, a PIR sensor for motion detection, a 4x4 keypad and RFID reader for user authentication, and output devices like an LCD display, LEDs, and a buzzer for notifications.

The software is written in embedded C and Assembly (before midterms), structured around modular code blocks that handle sensor interfacing, decision logic, and device control. The overall system was simulated and tested in Proteus software before deployment.

a. Problem

Modern homes generally face to face two significant challenges: high energy consumption and security measures. Household devices such as air conditioners frequently operate even when they are not needed, leading to unnecessary electricity usage. Meanwhile, many traditional home setups lack the ability to monitor intrusions or unauthorized access in real time, making them vulnerable to theft or safety risks.

Furthermore, the absence of automated monitoring and access control systems means that there is no effective way to respond to environmental changes or suspicious activities, which can compromise both energy efficiency and household safety.

b. Solution

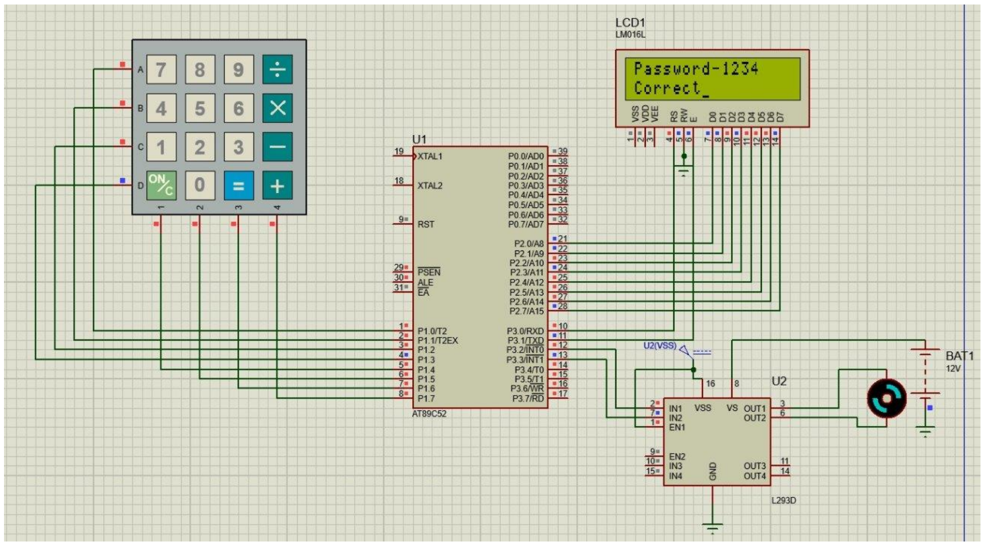
This project proposes a microcontroller-based smart home system that addresses both energy efficiency and security needs. The system utilizes sensors such as the LM35 temperature sensor to continuously monitor ambient temperature and automatically sends commands to turn the air conditioner on or off as needed, reducing unnecessary energy consumption. For enhanced security, a PIR motion sensor detects unauthorized movement and immediately triggers audible alarms or visual indicators.

Access to the home is further secured by integrating an RFID card reader and a keypad for password entry, ensuring that only authorized individuals can gain entry. All sensors and modules are managed by the AT89C51/AT89C52 microcontroller, which processes data in real time and coordinates device responses. The firmware is designed to be modular, enabling seamless integration and control of each component, while user feedback and notifications are displayed on an LCD screen.

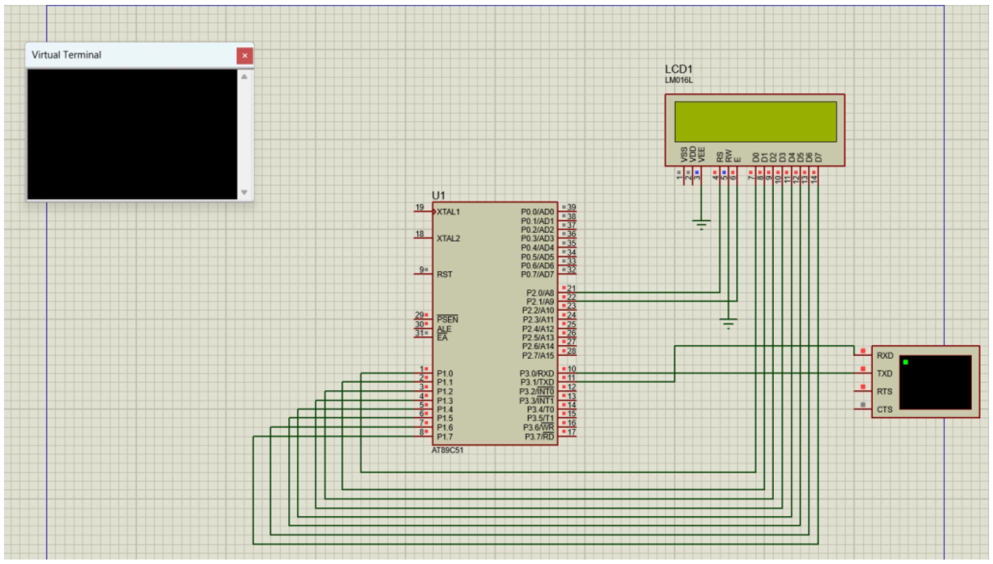
II. Design

The system consists of four independent microprocessor-based units:

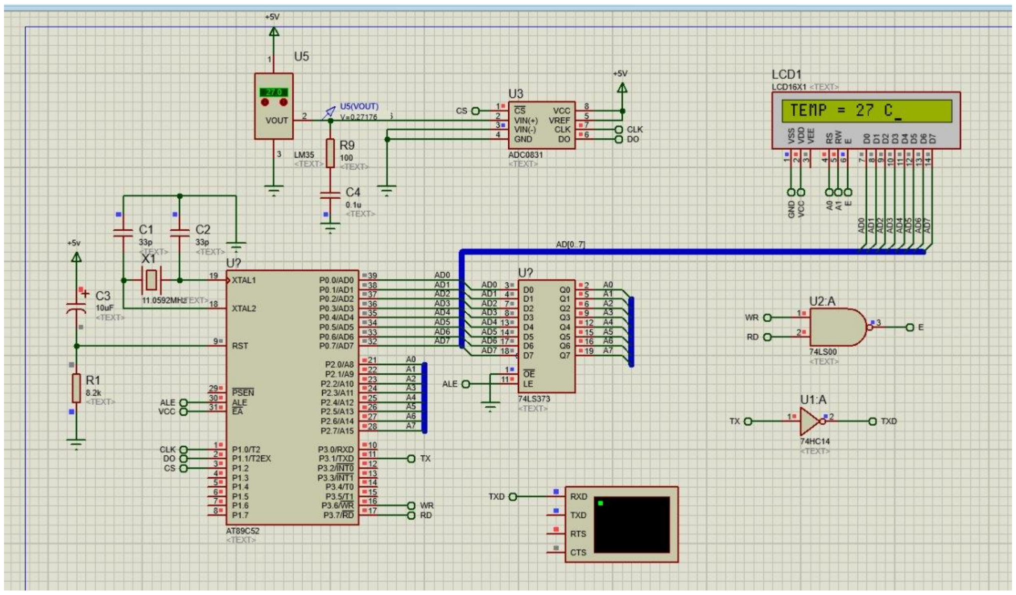
- **Keypad module:** Password entry for access control.



- **RFID module:** Card-based authentication.

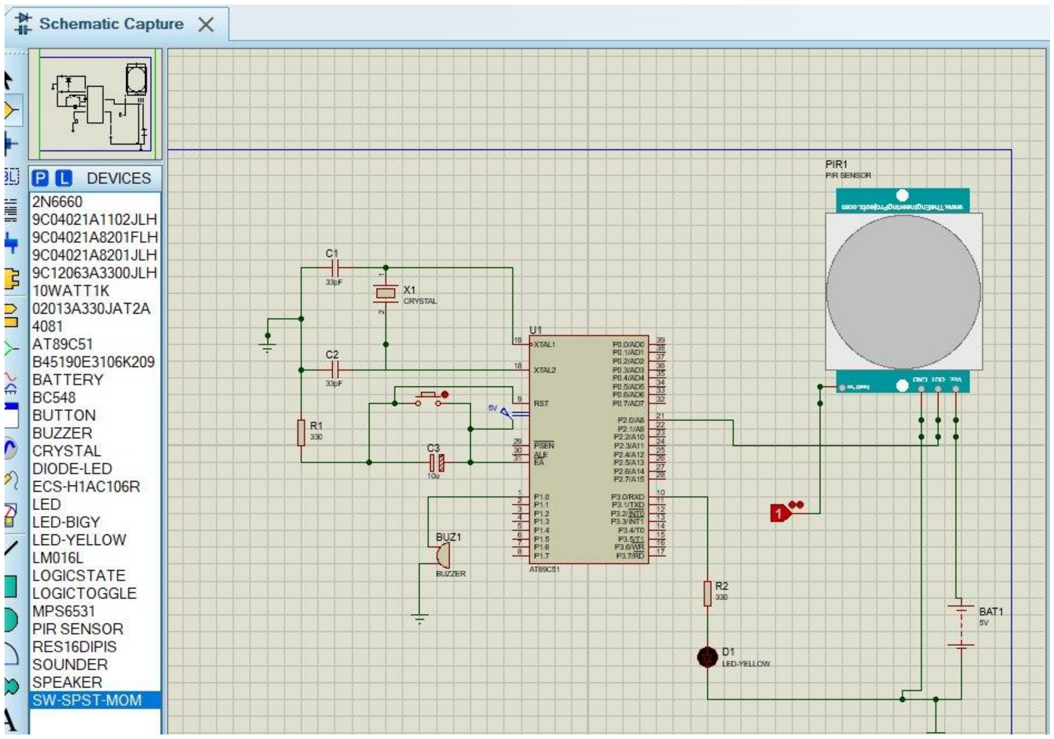


- **LM35 temperature module:** Real-time temperature monitoring.



- **PIR motion module:** Intrusion/motion detection.

Each module was designed and simulated in Proteus, and their firmware was developed in C and Assembly.



III. Achievements

a. Hardware

Microcontroller:

AT89C51/AT89C52 (8051 family), responsible for interfacing all sensors and outputs, executing control logic, and managing user interactions.

LM35 Temperature Sensor Circuit:

Measures ambient temperature. The microcontroller reads the analog value via ADC and displays the temperature in real-time on the LCD screen. If the temperature exceeds a set threshold, the system either activates a DC motor (fan simulation) or sends a notification/command to control the air conditioner (can also be simulated via UART terminal or a virtual relay).

PIR Motion Sensor Circuit:

Detects motion or presence in the environment. When motion is detected, the microcontroller triggers a **buzzer** or turns on a **LED** as an alert/alarm output. The status is also optionally shown on the LCD.

RFID Reader Circuit:

Reads RFID card data for authentication. When a valid card is scanned, the user's ID or access status is shown on the LCD. If an invalid card is detected, an error message appears and optionally a **buzzer** can be activated as a warning.

4x4 Keypad Circuit:

Allows users to enter passwords for access control. The keypad is connected directly to the microcontroller. Upon correct password entry, the system can activate a **relay**

(simulated by a **LED**) to unlock a door or grant access, and displays a confirmation on the LCD. Incorrect entries trigger a warning message and may also activate the buzzer.
LCD Display:
Provides real-time feedback and status messages for all modules, including temperature readings, access status, and error alerts.
Other Outputs:
LEDs: Used to indicate system status or successful authentication (e.g., access granted).
Buzzer: Used as an audible alarm or warning for motion detection, unauthorized access, or errors.
Relay/DC Motor: Used to control high-power devices (e.g., fan, door lock, or simulate AC control).

b. Firmware

i. C code for LM35

```
#include<reg51.h>
#define DATA_BUS P0
#define ADC_DATA P1

sbit WRadc = P3^3;
sbit RDadc = P3^4;
sbit RS = P2^0;
sbit RW = P2^1;
sbit EN = P2^2;
sbit BZ = P2^3;

typedef unsigned char uchar;
typedef unsigned int uint;

void LCD_init(void);
void lcdcmd(uchar value);
void lcddata(uchar value);
void printascii(uchar value);
void string(uchar ad, uchar * s);
void convert_display(uchar temp);

void delay(uint msec)
{
    uint i,j;
    for(i = 0; i < msec; i++)
        for(j = 0; j < 1275; j++);
}

void main()
{
    uchar temp = 0;
    LCD_init();
    while(1)
    {
        string(0x82, "TEMPERATURE:");

        RDadc = 1;
        WRadc = 0;
        delay(1);
        WRadc = 1;
        delay(1);
        RDadc = 0;
        delay(1);

        temp = ADC_DATA;
        convert_display(temp);
    }
}

void lcdcmd(uchar value)
{
    DATA_BUS = value;
    RS = 0;
    RW = 0;
    EN = 1;
    delay(1);
    EN = 0;
}

void lcddata(uchar value)
{
    DATA_BUS = value;
    RS = 1;
    RW = 0;
    EN = 1;
    delay(1);
    EN = 0;
}

void printascii(uchar value)
{
    value = value + 0x30;
    lcddata(value);
}

void LCD_init()
{
    lcdcmd(0x38);
    lcdcmd(0x0C);
    lcdcmd(0x06);
    lcdcmd(0x01);
    delay(1);
}

void string(uchar ad, uchar * s)
{
    lcdcmd(ad);
    while(*s > 0)
    {
        lcddata(*s++);
        delay(1);
    }
}

void convert_display(uchar temp)
{
    uchar h, t, u;
    lcdcmd(0xC6);
    if(temp >= 29)
        BZ = 1;
    else
```

```
                        BZ = 0;

                    h = temp / 100;
                    t = (temp / 10) % 10;
                    u = temp % 10;

                    printascii(h);
                    printascii(t);
                    printascii(u);
                    lcddata(0xDF);
                    lcddata('C');
                }
```

ii. Assembly equivalent of C code

```
; Varsayım: DATA_BUS = P0, ADC_DATA = P1 ; BZ = P2.3 (buzzer), LCD
control pins = P2.0 (RS), P2.1 (RW), P2.2 (EN)

   ORG 0000H           ; Program başlangıcı


START:

 ACALL LCD_INIT ; LCD başlatma rutini

MAIN_LOOP: ; --- "TEMPERATURE:" mesajı --- MOV DPTR, #MSG1 ACALL
LCD_STR

   ; --- ADC okuma işlemi ---
    SETB P3.4           ; RDadc = 1
    CLR  P3.3           ; WRadc = 0
    ACALL DELAY_1MS
    SETB P3.3           ; WRadc = 1
    ACALL DELAY_1MS
    CLR  P3.4           ; RDadc = 0
    ACALL DELAY_1MS

    ; --- ADC_DATA oku ---
    MOV A, P1           ; temp = ADC_DATA

    ; --- sıcaklık kontrolü ---
    CJNE A, #29, BELOW  ; temp >= 29?
    SETB P2.3           ; BZ = 1 (buzzer ON)
    SJMP CONT


BELOW:

 CLR P2.3 ; BZ = 0 (buzzer OFF) CONT:

 ; --- LCD’ye temp değerini yaz --- MOV R2, A ; R2 = temp MOV A, #0C6h
ACALL LCD_CMD

   ; --- temp’i rakamlara böl ve yaz ---
    MOV A, R2
    MOV B, #100
    DIV AB              ; h = temp / 100
    ACALL PRINT_ASCII   ; yüzler basamağı
    MOV A, B            ; kalan
    MOV B, #10
    DIV AB              ; t = temp / 10, u = temp % 10
    ACALL PRINT_ASCII   ; onlar basamağı
    MOV A, B
    ACALL PRINT_ASCII   ; birler basamağı
    MOV A, #0DFh
    ACALL LCD_DATA      ; derece işareti
    MOV A, #'C'
    ACALL LCD_DATA

    SJMP MAIN_LOOP


; ----------------------------------------- ; LCD ve diğer alt
rutinler burada tanımlanır ; -----------------------------------------
- LCD_INIT: MOV A, #038h ACALL LCD_CMD MOV A, #00Ch ACALL LCD_CMD MOV
A, #006h ACALL LCD_CMD MOV A, #001h ACALL LCD_CMD ACALL DELAY_1MS RET

LCD_CMD: ; (A register'daki komutu LCD'ye gönder) MOV P0, A CLR P2.0 ;
RS = 0 CLR P2.1 ; RW = 0 SETB P2.2 ; EN = 1 ACALL DELAY_1MS CLR P2.2 ;
EN = 0 RET
```

```
LCD_DATA: ; (A register'daki datayı LCD'ye gönder) MOV P0, A SETB P2.0
; RS = 1 CLR P2.1 ; RW = 0 SETB P2.2 ; EN = 1 ACALL DELAY_1MS CLR P2.2
; EN = 0 RET

PRINT_ASCII: ; (A register'daki değeri ASCII'ye çevirip LCD'ye bas)
ADD A, #30h ACALL LCD_DATA RET

LCD_STR: ; (DPTR ile gösterilen stringi LCD'ye yaz) CLR A NEXT_CHAR:
MOVC A, @A+DPTR JZ END_STR ACALL LCD_DATA INC DPTR SJMP NEXT_CHAR
END_STR: RET

DELAY_1MS: MOV R7, #250 D1: NOP DJNZ R7, D1 RET

MSG1: DB "TEMPERATURE:", 00h

                   END
```

iii. C code for RFID

```
#include <reg51.h>
#include <string.h>

sbit rs = P2^0;
sbit en = P2^1;
sfr ldata = 0x90;  // LCD veri hatti ? P1

unsigned char a[12];  // RFID ID verisi
unsigned int i;

void delay(unsigned int d) {
    unsigned int j;
    for(j = 0; j < d; j++);
}

void lcd_cmd(unsigned char cmd) {
    rs = 0;
    ldata = cmd;
    en = 1;
    delay(500);
    en = 0;
}
```

```
void lcd_data(unsigned char d) {
    rs = 1;
    ldata = d;
    en = 1;
    delay(500);
    en = 0;
}

void lcd_str(unsigned char *str) {
    while(*str) {
        lcd_data(*str++);
    }
}

void lcd_init() {
    lcd_cmd(0x38);  // 8-bit, 2-line
    lcd_cmd(0x0C);  // Display ON, Cursor OFF
    lcd_cmd(0x01);  // Clear display
    lcd_cmd(0x80);  // Start at first line
}

void clear_array() {
    for(i = 0; i < 12; i++) {
        a[i] = '\0';
    }
}

void main() {
    lcd_init();

    TMOD = 0x20;     // Timer1 mode2 - auto reload
    TH1 = 0xFD;      // 9600 baud rate
    SCON = 0x50;     // 8-bit UART mode
    TR1 = 1;         // Start Timer1

    lcd_str("RFID Ready");
    lcd_cmd(0xC0);
    lcd_str("Scan Your Card");
    while(1) {
        // 10 karakter RFID ID oku ve echo (geri
yazim) yap
        for(i = 0; i < 10; i++) {
            while(RI == 0);      // Veri gelene kadar
bekle
            a[i] = SBUF;         // Seri veriyi oku
            RI = 0;

            // Echo: karakteri terminale geri gönder
            SBUF = a[i];
            while(TI == 0);
            TI = 0;
        }

        a[10] = '\0';  // String sonlandirici

        lcd_cmd(0x01);  // LCD ekrani temizle
        lcd_cmd(0x80);

        if(strcmp(a, "10003B0CA7") == 0) {
            lcd_str("EMPLOYEE NO: 1");
        }
        else if(strcmp(a, "10003B0CAE") == 0) {
            lcd_str("EMPLOYEE NO: 2");
        }
        else {
            lcd_str("INVALID CARD");
        }

        lcd_cmd(0xC0);   // Degilse sunu dene:
lcd_cmd(0x94);

        lcd_str(a);  // Kart ID'yi LCD'de göster
        delay(30000);

        lcd_cmd(0x01);
        lcd_str("Scan Again...");
```

```
                    }
                }
```

iv. Assembly equivalent of C code

```
   ORG 0000H


; ------------------- ; LCD ve UART başlat ; ------------------- ACALL
LCD_INIT

   ; TMOD, TH1, SCON, TR1 ayarla
    MOV TMOD, #20h
    MOV TH1, #0FDh
    MOV SCON, #50h
    SETB TR1

    ; LCD: "RFID Ready"
    MOV DPTR, #MSG_READY
    ACALL LCD_STR
    MOV A, #0C0h
    ACALL LCD_CMD
    MOV DPTR, #MSG_SCAN
    ACALL LCD_STR


MAIN_LOOP:

   ; ------------------------------
    ; 10 karakter RFID ID oku
    ; ------------------------------
    MOV R2, #00h           ; i = 0


NEXT_CHAR: JNB RI, $ ; veri gelene kadar bekle MOV A, SBUF ; gelen
byte'ı oku CLR RI MOV R0, R2 ; dizinin adresi için MOV @A+DPTR, A ;
a[i] = A MOV SBUF, A ; echo WAIT_TI: JNB TI, WAIT_TI CLR TI

   INC R2
    CJNE R2, #0Ah, NEXT_CHAR ; 10 karakter okundu mu

    ; a[10] = '\0'
```

```
MOV A, #00h
MOV R0, #0Ah
MOV @A+DPTR, A

; ------------------------------
; LCD temizle
; ------------------------------
MOV A, #01h
ACALL LCD_CMD
MOV A, #80h
ACALL LCD_CMD

; ------------------------------
; RFID karşılaştır (pseudocode)
; ------------------------------
; NOT: Gerçek string karşılaştırma assembly'de
; döngü ile yapılır.
; if (strcmp(a, "10003B0CA7") == 0)
;     "EMPLOYEE NO: 1"
; else if (strcmp(a, "10003B0CAE") == 0)
;     "EMPLOYEE NO: 2"
; else
;     "INVALID CARD"
ACALL RFID_COMPARE     ; (Alt program, aşağıda örnek)

; LCD'ye ID yaz
MOV A, #0C0h
ACALL LCD_CMD
MOV DPTR, #a
ACALL LCD_STR

ACALL DELAY_LONG

; LCD: "Scan Again..."
MOV A, #01h
ACALL LCD_CMD
MOV DPTR, #MSG_AGAIN
ACALL LCD_STR
```

```
    SJMP MAIN_LOOP


; ---------------------------------------- ; LCD_INIT, LCD_CMD,
LCD_DATA, LCD_STR, ; DELAY, RFID_COMPARE alt programları burada ; ----
------------------------------------

; Mesaj sabitleri MSG_READY: DB "RFID Ready", 00h MSG_SCAN: DB "Scan
Your Card", 00h MSG_EMP1: DB "EMPLOYEE NO: 1", 00h MSG_EMP2: DB
"EMPLOYEE NO: 2", 00h MSG_INVLD: DB "INVALID CARD", 00h MSG_AGAIN: DB
"Scan Again...", 00h

; Diziler için alan ayır (örneğin, a: 12 byte) a: DS 12

                   END
```

v. C code for Keypad

```
#include<reg51.h> #include<string.h> sbit RS = P3^0; sbit EN = P3^1;
sbit IN1 =P3^2; sbit IN2 = P3^3; void delay(int a) { int i,j;
for(i=0;i<a;i++) for(j=0;j<255;j++); } void cmd(char cm) { P2 = cm; RS
= 0; EN = 1; delay(1); EN = 0; } void dat(char dt) { P2 = dt; RS = 1;
EN = 1; delay(1); EN = 0; }

void display(char *lcd) { while(*lcd != '\0') { dat(*lcd); lcd++; } }
void lcdint() { cmd(0x01); cmd(0x38); cmd(0x0E); cmd(0x80);

}

void main() { char pass[5] = "1234"; char pass2[5]; int i=0; char
*ptr; ptr = pass2; lcdint(); display("Password-"); pass2[4]='\0';

   while(1)
    {
    while(i<4)
{
    P1=0xFE;
    if(P1==0xEE)
    {
        *(ptr+i)='7';
        dat('7');
```

```
            delay(200);
            cmd(0x06);

            i++;

        }
        else if(P1==0xDE)
        {
            *(ptr+i)='8';
            dat('8');
            delay(200);
            cmd(0x06);

            i++;

        }
        else if(P1==0xBE)
        {
            *(ptr+i)='9';
            dat('9');
            delay(200);
            cmd(0x06);

            i++;
        }
        else if(P1==0x7E)
        {
            *(ptr+i)='/';
            dat('/');
            delay(200);
            cmd(0x06);

            i++;
        }


        P1=0xFD;
        if(P1==0xED)
        {
```

```
        *(ptr+i)='4';
        dat('4');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if(P1==0xDD)
    {
        *(ptr+i)='5';
        dat('5');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if(P1==0xBD)
    {
        *(ptr+i)='6';
        dat('6');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if (P1==0x7D)
    {
        *(ptr+i)='*';
        dat('*');
        delay(200);
        cmd(0x06);

        i++;
    }


    P1=0xFB;
    if(P1==0xEB)
    {
        *(ptr+i)='1';
        dat('1');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if(P1==0xDB)
    {
        *(ptr+i)='2';
        dat('2');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if(P1==0xBB)
    {
        *(ptr+i)='3';
        dat('3');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if(P1==0x7B)
    {
        *(ptr+i)='-';
        dat('-');
        delay(200);
        cmd(0x06);

        i++;
    }



    P1=0xF7;
    if(P1==0xE7)
    {
        *(ptr+i)='C';
        dat('C');
        delay(200);
        cmd(0x06);

        i++;

    }
    else if(P1==0xD7)
    {
        *(ptr+i)='0';
        dat('0');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if(P1==0xB7)
    {
        *(ptr+i)='=';
        dat('=');
        delay(200);
        cmd(0x06);

        i++;
    }
    else if(P1==0x77)
    {
        *(ptr+i)='+';
        dat('+');
        delay(200);
        cmd(0x06);

        i++;
    }
}

while(i==4) {
```

```
if ((strcmp(pass, pass2)) == 0) { cmd(0xC0); display("Correct"); IN1 =
1; IN2 = 0; delay(100);

} else { cmd(0xC0); display("Incorrect"); IN1 = 0; IN2 = 0;
delay(100); } } } }
```

vi. Assembly equivalent of C code

```
   ORG 0000H


START: ACALL LCD_INIT MOV DPTR, #MSG_PASS ACALL LCD_STR

MAIN_LOOP: MOV R2, #00h ; i = 0

READ_KEYPAD: ; --- Satır ve sütunları sırayla tara --- MOV P1, #0FEh ;
ilk satırı aktif et ; Burada: P1 okuma ve hangi tuşun basıldığını
bulma ; Örnek: P1 == 0xEE ise '7' yaz MOV A, P1 CJNE A, #0EEh, NEXT1
MOV R0, R2 MOV A, #'7' MOV @DPTR, A ; pass2[i] = '7' INC DPTR ACALL
LCD_DATA ; LCD'ye '7' yaz ACALL DELAY INC R2 ; i++ SJMP CHK_I

NEXT1: ; Diğer tuşlar için aynı mantıkla ; ... ; Tüm satır/sütun ve
tuş kontrolü yapılır, R2 artar

CHK_I: CJNE R2, #04h, READ_KEYPAD ; 4 karakter girilene kadar devam

; --- Şifre kontrolü --- MOV DPTR, #PASS1 ; orijinal şifre MOV R2,
#00h

CHECK_PASS: MOV A, @DPTR MOV R1, #pass2 ADD A, R2 MOV B, @R1 CJNE A,
B, WRONG INC DPTR INC R2 CJNE R2, #04h, CHECK_PASS

; --- Doğruysa --- MOV A, #0C0h ACALL LCD_CMD MOV DPTR, #MSG_CORR
ACALL LCD_STR SETB P3.2 ; IN1 = 1 CLR P3.3 ; IN2 = 0 ACALL DELAY_L
SJMP MAIN_LOOP

WRONG: MOV A, #0C0h ACALL LCD_CMD MOV DPTR, #MSG_INCORR ACALL LCD_STR
CLR P3.2 ; IN1 = 0 CLR P3.3 ; IN2 = 0 ACALL DELAY_L SJMP MAIN_LOOP

; ---------------------------------------- ; LCD_INIT, LCD_CMD,
LCD_DATA, LCD_STR, ; DELAY, DELAY_L, MSG_PASS, PASS1, MSG_CORR,
MSG_INCORR ; gibi alt program ve sabitler burada tanımlanır ; --------
-------------------------------

                END
```

vii. **C code for PIR**

```
#include <reg51.h>

// PIR sensörünün çıkışı P0.0’a, LED P1.0’a bağlanmalı
sbit PIR = P0^0;      // PIR sensör çıkışı
sbit LED = P1^0;      // Alarm LED’i

void Delay_ms(unsigned int ms) {
    unsigned int i, j;
    for (i = 0; i < ms; i++)
        for (j = 0; j < 112; j++);
}

void main(void) {
    LED = 0;            // Başta LED kapalı
    Delay_ms(3000);     // Sistem açılışında kısa
bekleme (sensör ısınması için)

    while (1) {
        if (PIR == 1) { // Hareket algılandıysa
            LED = 1;    // LED yak (alarm)
        } else {
            LED = 0;    // LED kapat
        }
    }
}
```

viii. **Assembly equivalent of C code**

```
        ORG 0000H

START:  CLR  P1.0         ; LED = 0 (başta kapalı)
        ACALL DELAY_3S    ; 3 saniye bekle
```

```
MAINLOOP:
        JB   P0.0, LED_ON ; Eğer PIR = 1 ise LED yak
        CLR  P1.0         ; PIR = 0 ise LED kapalı
        SJMP MAINLOOP

LED_ON: SETB P1.0         ; LED yak
        SJMP MAINLOOP

; 3 saniyelik delay (örnek):
DELAY_3S:
        MOV R7, #100
DELAY_LOOP1:
        MOV R6, #250
DELAY_LOOP2:
        MOV R5, #250
DELAY_LOOP3:
        NOP
        DJNZ R5, DELAY_LOOP3
        DJNZ R6, DELAY_LOOP2
        DJNZ R7, DELAY_LOOP1
        RET

        END
```

IV. Challenges and possible solutions

During this project, I faced several practical and technical challenges:

- **Simulation Software Limitations:**
  Since I used a cracked version of Proteus, I experienced restrictions—such as being unable to add a working PIR sensor library, or sometimes failing to compile certain codes. While I could simulate most circuits virtually, these limitations prevented a fully realistic testing environment.
- **Sensor and Library Compatibility Issues:**
  Since the 8051/AT89C51 family is an older microcontroller series, not every sensor module or library available today is compatible. Many modern sensors and libraries are designed for newer platforms like Arduino or STM32, making it difficult or sometimes

impossible to directly use them with 8051-based systems. This significantly limited the range of hardware and features I could integrate into my project.

- **Programming Hardware Constraints:**
  I was unable to upload the firmware to a real AT89C51/52 microcontroller, as the PROGISP programmer software did not run on my system. My antivirus flagged and blocked some files, making it impossible to complete a physical hardware test.
- **Lack of Modern Resources:**
  It was difficult to find up-to-date and detailed resources for the AT89C51 family. Most available tutorials and code examples were from 7-8 years ago, often in Hindi or on Indian YouTube channels, which made learning and troubleshooting time-consuming.
- **LDR (Light Dependent Resistor) Automation with 8051:**
  I also tried to design and simulate a light-sensitive automation circuit that uses an LDR sensor to turn off artificial lights when there is enough sunlight. However, I could not find enough up-to-date resources or example codes for implementing this with the AT89C51/AT89C52 microcontroller family. Most tutorials for LDR-based automation use Arduino or STM32, not the 8051 series, making it difficult to adapt solutions for my project.

V. Future plans

- In the future, I plan to upgrade my smart home project by integrating more advanced microcontrollers, such as STM32 or Arduino, to overcome compatibility and resource limitations faced with the AT89C51/AT89C52.
- I aim to further develop and implement additional features, including an LDR-based automatic lighting control system that turns off artificial lights when sufficient natural sunlight is detected.
- I also intend to use the components I have already purchased (sensors, modules, etc.) in my own home or future IoT projects.

VI. APPENDIX

Keypad.lst

```
C51 COMPILER V9.60.7.0 KEYPAD 06/12/2025 17:45:10 PAGE 1

C51 COMPILER V9.60.7.0, COMPILATION OF MODULE KEYPAD OBJECT MODULE
PLACED IN .\Objects\keypad.obj COMPILER INVOKED BY:
C:\Keil_v5\C51\BIN\C51.EXE keypad.c OPTIMIZE(8,SPEED) BROWSE DEBUG
OBJECTEXTEND PRINT(.\Listings\ke -ypad.lst) TABS(2)
OBJECT(.\Objects\keypad.obj)
```

```
line level source

1 #include<reg51.h> 2 #include<string.h> 3 sbit RS = P3^0; 4 sbit EN =
P3^1; 5 sbit IN1 =P3^2; 6 sbit IN2 = P3^3; 7 void delay(int a) 8 { 9 1
int i,j; 10 1 for(i=0;i<a;i++) 11 1 for(j=0;j<255;j++); 12 1 } 13 void
cmd(char cm) 14 { 15 1 P2 = cm; 16 1 RS = 0; 17 1 EN = 1; 18 1
delay(1); 19 1 EN = 0; 20 1 } 21 void dat(char dt) 22 { 23 1 P2 = dt;
24 1 RS = 1; 25 1 EN = 1; 26 1 delay(1); 27 1 EN = 0; 28 1 } 29

 30 void display(char *lcd) 31 { 32 1 while(*lcd != '\0') 33 1 { 34 2
dat(*lcd); 35 2 lcd++; 36 2 } 37 1 } 38 void lcdint() 39 { 40 1
cmd(0x01); 41 1 cmd(0x38); 42 1 cmd(0x0E); 43 1 cmd(0x80); 44 1

 45 1 } 46

 47 void main() 48 { 49 1 char pass[5] = "1234"; 50 1 char pass2[5];
51 1 int i=0; 52 1 char *ptr; 53 1 ptr = pass2; 54 1 lcdint(); C51
COMPILER V9.60.7.0 KEYPAD 06/12/2025 17:45:10 PAGE 2

55 1 display("Password-"); 56 1 pass2[4]='\0'; 57 1

 58 1 while(1) 59 1 { 60 2 while(i<4) 61 2 { 62 3 P1=0xFE; 63 3
if(P1==0xEE) 64 3 { 65 4 *(ptr+i)='7'; 66 4 dat('7'); 67 4 delay(200);
68 4 cmd(0x06); 69 4

 70 4 i++; 71 4

 72 4 } 73 3 else if(P1==0xDE) 74 3 { 75 4 *(ptr+i)='8'; 76 4
dat('8'); 77 4 delay(200); 78 4 cmd(0x06); 79 4

 80 4 i++; 81 4

 82 4 } 83 3 else if(P1==0xBE) 84 3 { 85 4 *(ptr+i)='9'; 86 4
dat('9'); 87 4 delay(200); 88 4 cmd(0x06); 89 4

 90 4 i++; 91 4 } 92 3 else if(P1==0x7E) 93 3 { 94 4 *(ptr+i)='/'; 95
4 dat('/'); 96 4 delay(200); 97 4 cmd(0x06); 98 4

 99 4 i++; 100 4 } 101 3

 102 3
 103 3 P1=0xFD; 104 3 if(P1==0xED) 105 3 { 106 4 *(ptr+i)='4'; 107 4
dat('4'); 108 4 delay(200); 109 4 cmd(0x06); 110 4

 111 4 i++; 112 4 } 113 3 else if(P1==0xDD) 114 3 { 115 4
*(ptr+i)='5'; 116 4 dat('5'); C51 COMPILER V9.60.7.0 KEYPAD 06/12/2025
17:45:10 PAGE 3

117 4 delay(200); 118 4 cmd(0x06); 119 4

 120 4 i++; 121 4 } 122 3 else if(P1==0xBD) 123 3 { 124 4 (ptr+i)='6';
125 4 dat('6'); 126 4 delay(200); 127 4 cmd(0x06); 128 4

 129 4 i++; 130 4 } 131 3 else if (P1==0x7D) 132 3 { 133 4 (ptr+i)='';
134 4 dat(''); 135 4 delay(200); 136 4 cmd(0x06); 137 4

 138 4 i++; 139 4 } 140 3

 141 3

 142 3 P1=0xFB; 143 3 if(P1==0xEB) 144 3 { 145 4 *(ptr+i)='1'; 146 4
dat('1'); 147 4 delay(200); 148 4 cmd(0x06); 149 4

 150 4 i++; 151 4 } 152 3 else if(P1==0xDB) 153 3 { 154 4
*(ptr+i)='2'; 155 4 dat('2'); 156 4 delay(200); 157 4 cmd(0x06); 158 4

 159 4 i++; 160 4 } 161 3 else if(P1==0xBB) 162 3 { 163 4
*(ptr+i)='3'; 164 4 dat('3'); 165 4 delay(200); 166 4 cmd(0x06); 167 4

 168 4 i++; 169 4 } 170 3 else if(P1==0x7B) 171 3 { 172 4 *(ptr+i)='-
'; 173 4 dat('-'); 174 4 delay(200); 175 4 cmd(0x06); 176 4

 177 4 i++; 178 4 } C51 COMPILER V9.60.7.0 KEYPAD 06/12/2025 17:45:10
PAGE 4

179 3

 180 3

 181 3

 182 3 P1=0xF7; 183 3 if(P1==0xE7) 184 3 { 185 4 *(ptr+i)='C'; 186 4
dat('C'); 187 4 delay(200); 188 4 cmd(0x06); 189 4

 190 4 i++; 191 4
 192 4 } 193 3 else if(P1==0xD7) 194 3 { 195 4 *(ptr+i)='0'; 196 4
dat('0'); 197 4 delay(200); 198 4 cmd(0x06); 199 4

 200 4 i++; 201 4 } 202 3 else if(P1==0xB7) 203 3 { 204 4
*(ptr+i)='='; 205 4 dat('='); 206 4 delay(200); 207 4 cmd(0x06); 208 4

 209 4 i++; 210 4 } 211 3 else if(P1==0x77) 212 3 { 213 4
*(ptr+i)='+'; 214 4 dat('+'); 215 4 delay(200); 216 4 cmd(0x06); 217 4

 218 4 i++; 219 4 } 220 3 } 221 2

 222 2 while(i==4) 223 2 { 224 3

 225 3 if ((strcmp(pass, pass2)) == 0) 226 3 { 227 4 cmd(0xC0); 228 4
display("Correct"); 229 4 IN1 = 1; 230 4 IN2 = 0; 231 4 delay(100);
232 4 } 233 3 else 234 3 { 235 4 cmd(0xC0); 236 4
display("Incorrect"); 237 4 IN1 = 0; 238 4 IN2 = 0; 239 4 delay(100);
240 4 } C51 COMPILER V9.60.7.0 KEYPAD 06/12/2025 17:45:10 PAGE 5

241 3 } 242 2 } 243 1 }

MODULE INFORMATION: STATIC OVERLAYABLE CODE SIZE = 883 ---- CONSTANT
SIZE = 33 ---- XDATA SIZE = ---- ---- PDATA SIZE = ---- ---- DATA SIZE
= ---- 18 IDATA SIZE = ---- ---- BIT SIZE = ---- ---- END OF MODULE
INFORMATION.

C51 COMPILATION COMPLETE. 0 WARNING(S), 0 ERROR(S)
```

Rfid.lst

```
     C51 COMPILER V9.60.7.0   RFID
06/14/2025 15:33:30 PAGE 1



     C51 COMPILER V9.60.7.0, COMPILATION OF MODULE RFID

     OBJECT MODULE PLACED IN .\Objects\rfid.obj

     COMPILER INVOKED BY: C:\Keil_v5\C51\BIN\C51.EXE ..\rfid.c OMF2
OPTIMIZE(8,SPEED) BROWSE DEBUG PRINT(.\Listings\rfid.lst)
```

```
                - TABS(2) OBJECT(.\Objects\rfid.obj)

line level    source

   1          #include <reg51.h>
   2          #include <string.h>
   3          
   4          sbit rs = P2^0;
   5          sbit en = P2^1;
   6          sfr ldata = 0x90;  // LCD veri hatti ? P1
   7          
   8          unsigned char a[12];  // RFID ID verisi
   9          unsigned int i;
  10          
  11          void delay(unsigned int d) {
  12   1          unsigned int j;
  13   1          for(j = 0; j < d; j++);
  14   1      }
  15          
  16          void lcd_cmd(unsigned char cmd) {
  17   1          rs = 0;
  18   1          ldata = cmd;
  19   1          en = 1;
  20   1          delay(500);
  21   1          en = 0;
  22   1      }
  23          
```

```
       void lcd_data(unsigned char d) {
  1        rs = 1;
  1        ldata = d;
  1        en = 1;
  1        delay(500);
  1        en = 0;
  1    }

       void lcd_str(unsigned char *str) {
  1        while(*str) {
  2            lcd_data(*str++);
  2        }
  1    }

       void lcd_init() {
  1        lcd_cmd(0x38);  // 8-bit, 2-line
  1        lcd_cmd(0x0C);  // Display ON, Cursor OFF
  1        lcd_cmd(0x01);  // Clear display
  1        lcd_cmd(0x80);  // Start at first line
  1    }

       void clear_array() {
  1        for(i = 0; i < 12; i++) {
  2            a[i] = '\0';
  2        }
  1    }

```

```
        51          void main() {
        52   1          lcd_init();
        53   1
        54   1          TMOD = 0x20;     // Timer1 mode2 - auto reload
     C51 COMPILER V9.60.7.0   RFID
06/14/2025 15:33:30 PAGE 2


        55   1          TH1 = 0xFD;      // 9600 baud rate
        56   1          SCON = 0x50;     // 8-bit UART mode
        57   1          TR1 = 1;         // Start Timer1
        58   1
        59   1          lcd_str("RFID Ready");
        60   1          lcd_cmd(0xC0);
        61   1          lcd_str("Scan Your Card");
        62   1
        63   1          while(1) {
        64   2              // 10 karakter RFID ID oku ve echo (geri
yazim) yap
        65   2              for(i = 0; i < 10; i++) {
        66   3                  while(RI == 0);      // Veri gelene
kadar bekle
        67   3                  a[i] = SBUF;         // Seri veriyi oku
        68   3                  RI = 0;
        69   3
        70   3                  // Echo: karakteri terminale geri
gönder
        71   3                  SBUF = a[i];
        72   3                  while(TI == 0);
3               TI = 0;
3           }
2
2           a[10] = '\0';  // String sonlandirici
2
2           lcd_cmd(0x01);  // LCD ekrani temizle
2           lcd_cmd(0x80);
2
2           if(strcmp(a, "10003B0CA7") == 0) {
3               lcd_str("EMPLOYEE NO: 1");
3           }
2           else if(strcmp(a, "10003B0CAE") == 0) {
3               lcd_str("EMPLOYEE NO: 2");
3           }
2           else {
3               lcd_str("INVALID CARD");
3           }
2
2           lcd_cmd(0xC0);   // Degilse sunu dene:
2           lcd_cmd(0x94);
2
2           lcd_str(a);  // Kart ID'yi LCD'de göster
2           delay(30000);
2
2           lcd_cmd(0x01);
2           lcd_str("Scan Again...");
2       }
```

```
100   1      }



MODULE INFORMATION:   STATIC OVERLAYABLE
   CODE SIZE        =    322    ----
   CONSTANT SIZE    =    105    ----
   XDATA SIZE       =   ----    ----
   PDATA SIZE       =   ----    ----
   DATA SIZE        =     14       3
   IDATA SIZE       =   ----    ----
   BIT SIZE         =   ----    ----
   EDATA SIZE       =   ----    ----
   HDATA SIZE       =   ----    ----
   XDATA CONST SIZE =   ----    ----
   FAR CONST SIZE   =   ----    ----
END OF MODULE INFORMATION.



C51 COMPILATION COMPLETE.  0 WARNING(S),  0 ERROR(S)
```

Lm35.lst

```
C51 COMPILER V9.60.7.0   MAIN
06/05/2025 00:55:21 PAGE 1



C51 COMPILER V9.60.7.0, COMPILATION OF MODULE MAIN
OBJECT MODULE PLACED IN .\Objects\main.obj
```

```
      COMPILER INVOKED BY: C:\Keil_v5\C51\BIN\C51.EXE main.c
OPTIMIZE(8,SPEED) BROWSE DEBUG OBJECTEXTEND PRINT(.\Listings\main
                          -.lst) TABS(2) OBJECT(.\Objects\main.obj)


      line level    source

#include<reg51.h>
#define DATA_BUS P0
#define ADC_DATA P1

sbit WRadc = P3^3;
sbit RDadc = P3^4;
sbit RS = P2^0;
sbit RW = P2^1;
sbit EN = P2^2;
sbit BZ = P2^3;

typedef unsigned char uchar;
typedef unsigned int uint;

void LCD_init(void);
void lcdcmd(uchar value);
void lcddata(uchar value);
void printascii(uchar value);
void string(uchar ad, uchar * s);
void convert_display(uchar temp);

```

```
          void delay(uint msec)
          {
   1          uint i,j;
   1          for(i = 0; i < msec; i++)
   1              for(j = 0; j < 1275; j++);
   1      }

          void main()
          {
   1          uchar temp = 0;
   1          LCD_init();
   1          while(1)
   1          {
   2              string(0x82, "TEMPERATURE:");
   2
   2              RDadc = 1;
   2              WRadc = 0;
   2              delay(1);
   2              WRadc = 1;
   2              delay(1);
   2              RDadc = 0;
   2              delay(1);
   2
   2              temp = ADC_DATA;
   2              convert_display(temp);
   2          }
   1      }
```

```

        void lcdcmd(uchar value)
        {
 1          DATA_BUS = value;
 1          RS = 0;
 1          RW = 0;
     C51 COMPILER V9.60.7.0   MAIN
06/05/2025 00:55:21 PAGE 2


 1          EN = 1;
 1          delay(1);
 1          EN = 0;
 1      }

        void lcddata(uchar value)
        {
 1          DATA_BUS = value;
 1          RS = 1;
 1          RW = 0;
 1          EN = 1;
 1          delay(1);
 1          EN = 0;
 1      }

        void printascii(uchar value)
        {
 1          value = value + 0x30;
```

```
1           lcddata(value);
1       }

        void LCD_init()
        {
1           lcdcmd(0x38);
1           lcdcmd(0x0C);
1           lcdcmd(0x06);
1           lcdcmd(0x01);
1           delay(1);
1       }

        void string(uchar ad, uchar * s)
        {
1           lcdcmd(ad);
1           while(*s > 0)
1           {
2               lcddata(*s++);
2               delay(1);
2           }
1       }

        void convert_display(uchar temp)
        {
1           uchar h, t, u;
1           lcdcmd(0xC6);
1           if(temp >= 29)
```

```
    100   1               BZ = 1;
    101   1           else
    102   1               BZ = 0;
    103   1
    104   1           h = temp / 100;
    105   1           t = (temp / 10) % 10;
    106   1           u = temp % 10;
    107   1
    108   1           printascii(h);
    109   1           printascii(t);
    110   1           printascii(u);
    111   1           lcddata(0xDF);
    112   1           lcddata('C');
    113   1       }


    MODULE INFORMATION:   STATIC OVERLAYABLE
    C51 COMPILER V9.60.7.0   MAIN
06/05/2025 00:55:21 PAGE 3


       CODE SIZE        =    253    ----
       CONSTANT SIZE    =     13    ----
       XDATA SIZE       =   ----    ----
       PDATA SIZE       =   ----    ----
       DATA SIZE        =   ----       5
       IDATA SIZE       =   ----    ----
       BIT SIZE         =   ----    ----
```

```
END OF MODULE INFORMATION.


C51 COMPILATION COMPLETE.  0 WARNING(S),  0 ERROR(S)
```

Pir.lst

```
C51 COMPILER V9.60.7.0   PIR
06/10/2025 01:42:24 PAGE 1


C51 COMPILER V9.60.7.0, COMPILATION OF MODULE PIR
OBJECT MODULE PLACED IN .\Objects\PIR.obj
COMPILER INVOKED BY: C:\Keil_v5\C51\BIN\C51.EXE PIR.c OMF2
OPTIMIZE(8,SPEED) BROWSE DEBUG PRINT(.\Listings\PIR.lst) TABS
                          -(2) OBJECT(.\Objects\PIR.obj)

line level    source

   1          #include <reg51.h>
   2
   3          sbit Motion = P0^0;
   4          sbit LED    = P1^0;
   5
   6          void Delay_ms(unsigned int ms) {
   7   1          unsigned int i, j;
   8   1          for (i = 0; i < ms; i++)
   9   1              for (j = 0; j < 112; j++);
```

```
  10   1      }
  11
  12          void main(void) {
  13   1          P1 = 0x00;
  14   1          Delay_ms(3000);
  15   1
  16   1          while (1) {
  17   2              if (Motion == 1) {
  18   3                  LED = 0;
  19   3              } else {
  20   3                  LED = 1;
  21   3              }
  22   2          }
  23   1      }


MODULE INFORMATION:   STATIC OVERLAYABLE
   CODE SIZE        =     53    ----
   CONSTANT SIZE    =   ----    ----
   XDATA SIZE       =   ----    ----
   PDATA SIZE       =   ----    ----
   DATA SIZE        =   ----    ----
   IDATA SIZE       =   ----    ----
   BIT SIZE         =   ----    ----
   EDATA SIZE       =   ----    ----
   HDATA SIZE       =   ----    ----
   XDATA CONST SIZE =   ----    ----
   FAR CONST SIZE   =   ----    ----
END OF MODULE INFORMATION.



C51 COMPILATION COMPLETE.  0 WARNING(S),  0 ERROR(S)
```