

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.

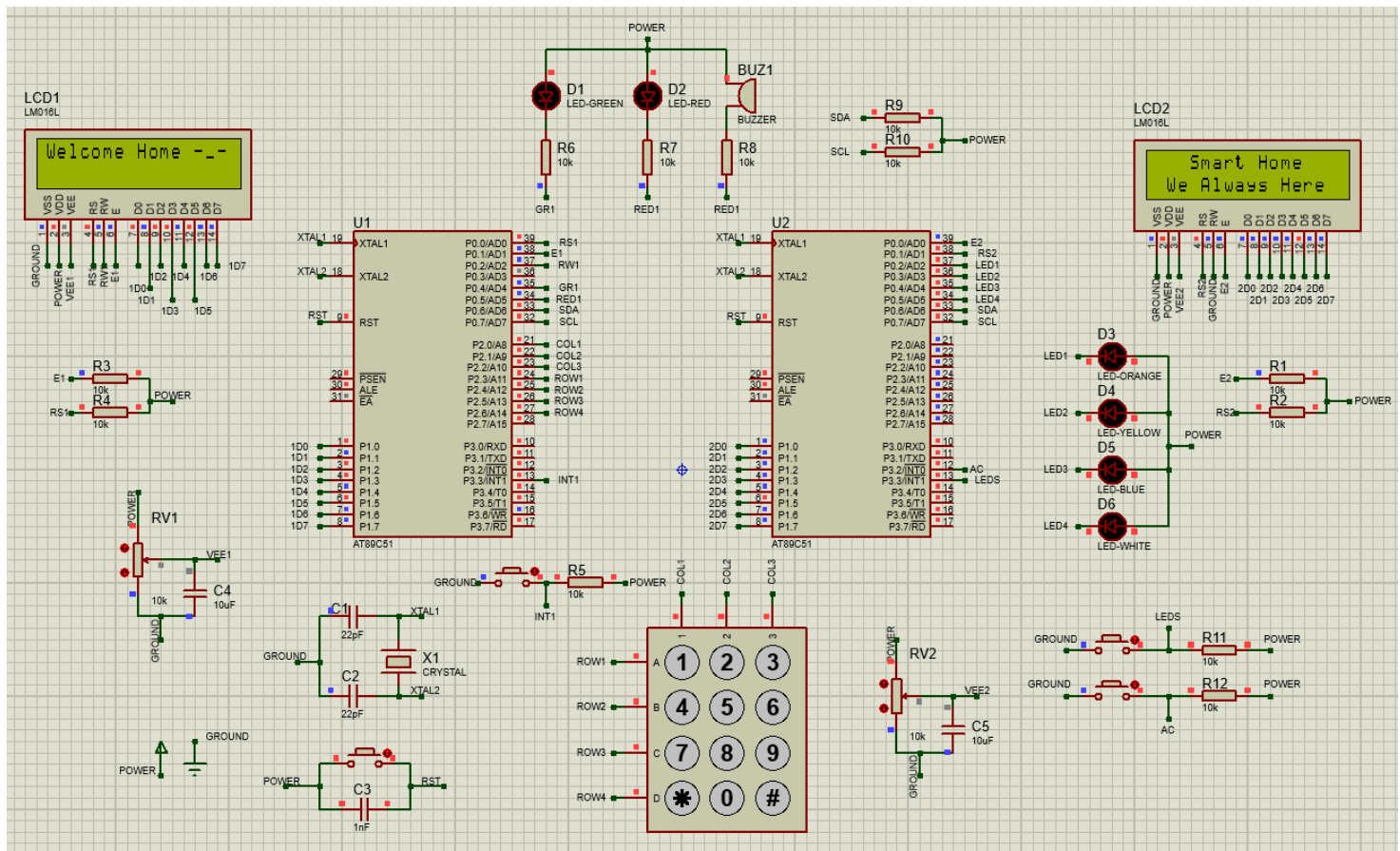
The function used here is:

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

2.5 Why I²C is a Good Fit

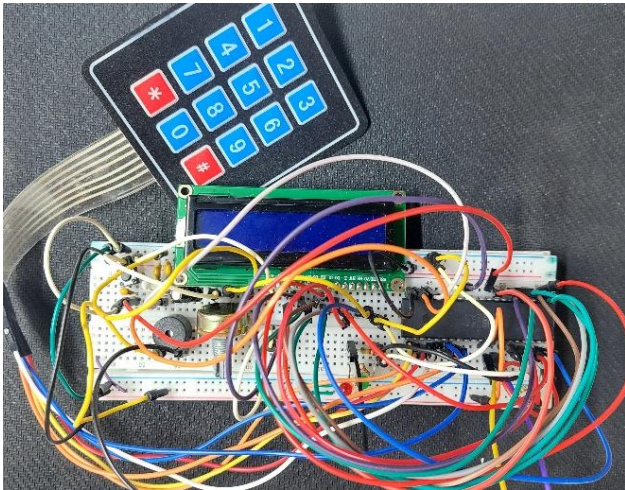
- **Only 2 wires** needed between the microcontrollers.
- Easy to implement in software.
- Allows for **simple, fast, and reliable communication**.
- Can be extended later to send more commands or data if needed.

3. Simulation Circuit Diagram:

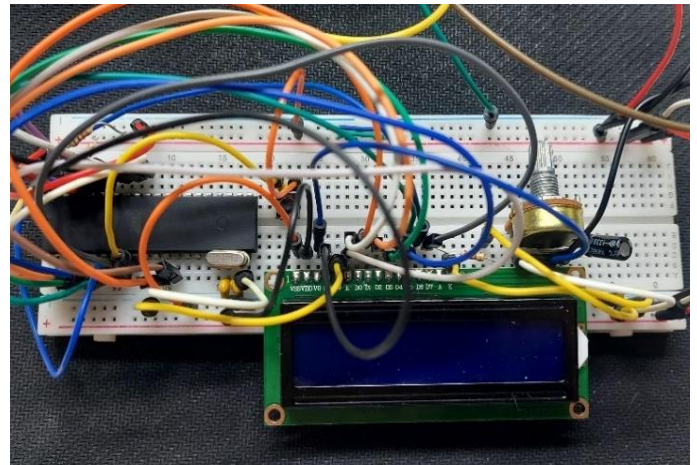


4. Circuit on real life:

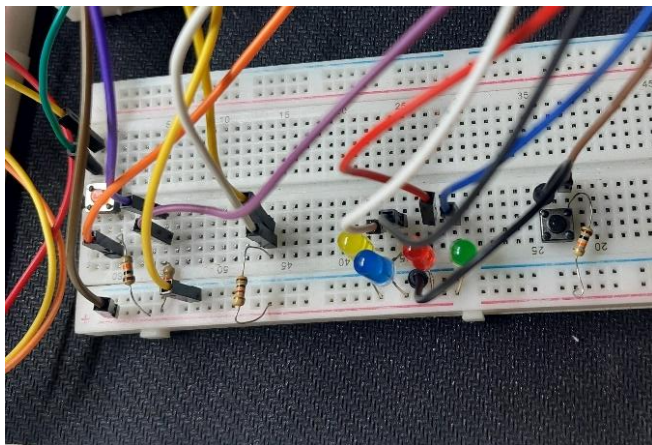
First Microcontroller Breadboard



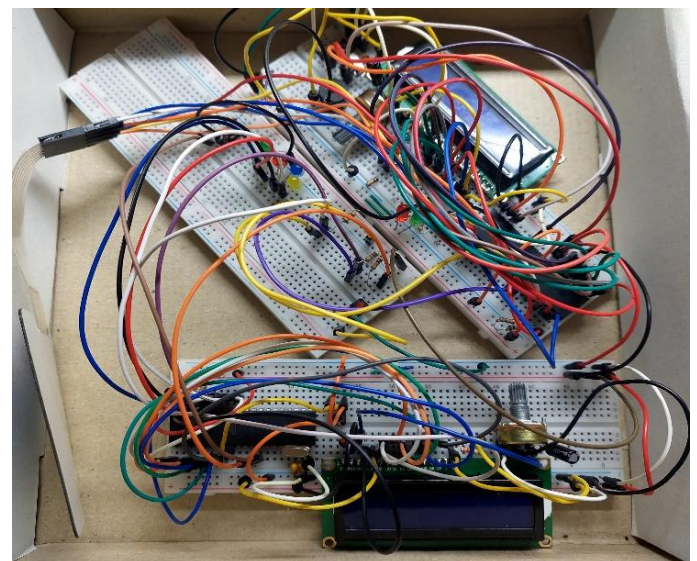
Second Microcontroller Breadboard



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

```

1  #include <reg51.h>
2  #include "lcd.h"
3  #include "keypad.h"
4  #include "leds.h"
5  #include "delay.h"
6  #include "password.h"
7  #include "function.h"
8  #include "i2c.h"
9  #include "button.h"
10
11 unsigned char correct_pass_flag = 0;
12 unsigned char reEnter_pass_flag = 0;
13
14 void main() {
15     unsigned char i;
16     static bit welcome_shown = 0;
17
18     LCD_init();
19     I2C_init();
20     LED_setRed(LED_OFF);
21     LED_setGreen(LED_OFF);
22
23     while (1) {
24         if (Button_readStart() == BUTTON_PRESSED) {
25             welcome_shown = 0;
26
27             LCD_clear();
28             LCD_writeText("Enter Password:");
29             LCD_setCursor(1, 0);
30             Password_inputFromUser();
31             Password_checkCorrect();
32
33             if (correct_pass_flag == 0) {
34                 LED_setGreen(LED_ON);
35                 LED_setRed(LED_OFF);
36                 num_ofCorrect_Password = 3;
37
38                 Password_successAction();
39                 I2C_start();
40                 I2C_write('S');
41                 I2C_stop();
42                 delay_ms(300);
43             } else {
44                 num_ofCorrect_Password--;
45                 LED_setRed(LED_ON);
46                 LED_setGreen(LED_OFF);
47
48                 while (num_ofCorrect_Password != 0) {
49                     LCD_clear();
50                     LCD_writeText("WRONG PASS");
51                     delay_ms(300);
52
53                     LCD_clear();
54                     LCD_writeText("Attempts Left:");
55                     LCD_writeChar(num_ofCorrect_Password + '0');
56                     delay_ms(300);
57
58                     LCD_clear();
59                     LCD_writeText("Enter Password:");
60                     LCD_setCursor(1, 0);
61                     Password_inputFromUser();
62                     Password_checkCorrect();
63
64                     if (correct_pass_flag == 1) {
65                         num_ofCorrect_Password--;
66                         if (num_ofCorrect_Password == 1) {
67                             LED_setRed(LED_ON);
68                         }
69                         delay_ms(100);
70                     } else if (correct_pass_flag == 0) {
71                         reEnter_pass_flag = 1;
72                         num_ofCorrect_Password = 3;
73                         break;
74                     }
75                 }
76             }
77         }
78     }
79 }

```



```

78 if (reEnter_pass_flag == 1) {
79     reEnter_pass_flag = 0;
80     LED_setGreen(LED_ON);
81     LED_setRed(LED_OFF);
82
83     Password_successAction();
84     I2C_start();
85     I2C_write('S');
86     I2C_stop();
87     delay_ms(300);
88 }
89
90 if (num_ofCorrect_Password == 0) {
91     LCD_clear();
92     LCD_writeText("ACCESS DENIED");
93     LCD_setCursor(1, 0);
94     LCD_writeText("INTRUDER ALERT!");
95     delay_ms(1000);
96
97     for (i = 0; i < 6; i++) {
98         if (i % 2 == 0) {
99             LED_setRed(LED_ON);
100             LED_setGreen(LED_OFF);
101         } else {
102             LED_setRed(LED_OFF);
103             LED_setGreen(LED_ON);
104         }
105         delay_ms(300);
106     }
107
108     while (1); // System locked; manual reset needed
109 }
110 } else if (!welcome_shown) {
111     LCD_clear();
112     LCD_writeText("Welcome Home _ _");
113     welcome_shown = 1;
114 }
115 }
116 }
117 }

```

• Function code

```

1 #ifndef FUNCTION_H
2 #define FUNCTION_H
3
4 // Function Prototypes
5 void Password_inputFromUser(void);
6 void Password_checkCorrect(void);
7
8 #endif /* FUNCTION_H */
9
10 #include "function.h"
11 #include "lcd.h"
12 #include "keypad.h"
13 #include "delay.h"
14 #include "password.h"
15
16 extern unsigned char correct_pass_flag;
17
18 void Password_inputFromUser(void) {
19     unsigned char i = 0, key;
20     for (i = 0; i < 4; i++) entered_pass[i] = 0;
21
22     i = 0;
23     while (1) {
24         key = Keypad_getKey();
25         if (key == '#') break;
26
27         if (key >= '0' && key <= '9' && i < 4) {
28             entered_pass[i] = key;
29             LCD_writeChar(entered_pass[i]);
30             i++;
31         } else if (key == '*' && i > 0) {
32             i--;
33             entered_pass[i] = 0;
34             LCD_setCursor(1, i);
35             LCD_writeChar(' ');
36             LCD_setCursor(1, i);
37         }
38     }
39     delay_ms(50);
40 }
41
42 while (Keypad_getKey() != '#');
43 }
44
45 void Password_checkCorrect(void) {
46     unsigned char i;
47     correct_pass_flag = 0;
48     for (i = 0; i < 4; i++) {
49         if (entered_pass[i] != saved_pass[i])
50             correct_pass_flag = 1;
51     }
52 }
53 }
54 }

```



• Password code

```

1 #ifndef PASSWORD_H
2 #define PASSWORD_H
3
4 // Global Variables (extern)
5 extern unsigned char saved_pass[5];
6 extern unsigned char entered_pass[5];
7 extern unsigned char num_ofCorrect_Password;
8
9 // Function Prototypes
10 void Password_successAction(void);
11
12 #endif /* PASSWORD H */
13
14 #include "password.h"
15 #include "lcd.h"
16 #include "leds.h"
17 #include "delay.h"
18
19 // Define real memory for globals
20 unsigned char saved_pass[5] = {'1', '2', '3', '4', '\0'};
21 unsigned char entered_pass[5];
22 unsigned char num_ofCorrect_Password = 3;
23
24 void Password_successAction(void) {
25     LCD_clear();
26     LCD_writeText("Open");
27     delay_ms(1000);
28
29     LCD_clear();
30     LCD_writeText("Close");
31     delay_ms(500);
32
33     LED_setGreen(LED_OFF);
34     LCD_clear();
35 }

```

• I2C code

```

1 #ifndef I2C_H
2 #define I2C_H
3
4 #include <reg51.h>
5
6 sbit SDA = P0^6;
7 sbit SCL = P0^7;
8
9
10 void I2C_init(void);
11 void I2C_start(void);
12 void I2C_stop(void);
13 void I2C_write(unsigned char byte);
14 bit I2C_WaitAck(void);
15
16 #endif
17
18 #include "i2c.h"
19 #include "delay.h"
20
21 void I2C_init(void) {
22     SDA = 1;
23     SCL = 1;
24 }
25
26 void I2C_Start(void) {
27     SDA = 1; SCL = 1;
28     delay_ms(1);
29     SDA = 0;
30     delay_ms(1);
31     SCL = 0;
32 }
33
34 void I2C_Stop(void) {
35     SCL = 0; SDA = 0;
36     delay_ms(1);
37     SCL = 1;
38     SDA = 1;
39     delay_ms(1);
40 }
41
42 bit I2C_WaitAck(void) {
43     SDA = 1;
44     SCL = 1;
45     delay_ms(1);
46     if (SDA) {
47         SCL = 0;
48         return 0;
49     }
50     SCL = 0;

```



```

34     return 1;
35 }
36
37 void I2C_Write(unsigned char dat) {
38     unsigned char i;
39     for (i = 0; i < 8; i++) {
40         SDA = (dat & 0x80);
41         SCL = 1;
42         delay_ms(1);
43         SCL = 0;
44         dat <<= 1;
45     }
46     I2C_WaitAck();
47 }

```

• Button code

```

1 #ifndef BUTTON_H
2 #define BUTTON_H
3
4 #include <reg51.h>
5
6 // Define START button pin properly
7 sbit START_BUTTON_PIN = P3^3;
8
9 // Button states
10 #define BUTTON_PRESSED 1
11 #define BUTTON_RELEASED 0
12
13 // Function Prototype
14 unsigned char Button_readStart(void);
15
16 #endif
17
18 #include "button.h"
19 #include "delay.h"
20
21 unsigned char Button_readStart(void) {
22     if (START_BUTTON_PIN == 0) {
23         delay_ms(20);
24         if (START_BUTTON_PIN == 0) {
25             while (START_BUTTON_PIN == 0);
26             return BUTTON_PRESSED;
27         }
28     }
29     return BUTTON_RELEASED;
30 }

```

• LEDs code

```

1 #ifndef LEDS_H
2 #define LEDS_H
3
4 #include <reg51.h>
5
6 // LED Pin Definitions
7 sbit LED_RED = P0^5;
8 sbit LED_GREEN = P0^4;
9
10 // LED States
11 #define LED_ON 0
12 #define LED_OFF 1
13
14 // Function Prototypes
15 void LED_setRed(unsigned char state);
16 void LED_setGreen(unsigned char state);
17
18 #endif /* LEDS_H */
19
20 #include "leds.h"
21
22 void LED_setRed(unsigned char state) {
23     LED_RED = state;
24 }
25
26 void LED_setGreen(unsigned char state) {
27     LED_GREEN = state;
28 }

```

• Delay code

```

1 #ifndef DELAY_H
2 #define DELAY_H
3
4 void delay_ms(unsigned int ms);
5
6 #endif
7
8 #include "delay.h"
9 #include <reg51.h>
10
11 void delay_ms(unsigned int ms) {
12     unsigned int i, j;
13     for (i = 0; i < ms; i++)
14         for (j = 0; j < 1275; j++);
15 }

```



• LCD code

```
1 #ifndef LCD_H
2 #define LCD_H
3
4 #include <reg51.h>
5 #include "delay.h"
6
7 // LCD Control Pins using sbit
8 sbit LCD_RS = P0^0;
9 sbit LCD_E = P0^1;
10 sbit LCD_RW = P0^2;
11
12 // LCD Data Port
13 #define LCD_DATA_PORT P1
14
15 // LCD Command Constants
16 #define LCD_CLEAR_COMMAND 0x01
17 #define LCD_GO_TO_HOME 0x02
18 #define LCD_TWO_LINES_EIGHT_BITS_MODE 0x38
19 #define LCD_CURSOR_OFF 0x0C
20 #define LCD_CURSOR_ON 0x0E
21 #define LCD_SET_CURSOR_LOCATION 0x80
22
23 // Function Prototypes
24 void LCD_init(void);
25 void LCD_sendCommand(unsigned char command);
26 void LCD_writeChar(unsigned char character);
27 void LCD_writeText(const char *text);
28 void LCD_clear(void);
29 void LCD_setCursor(unsigned char row, unsigned char col);
30
31 #endif /* LCD_H */
32
33 #include <reg51.h>
34 #include "delay.h"
35 #include "lcd.h"
36
37 void LCD_init(void) {
38     LCD_RS = 0;
39     LCD_E = 0;
40     LCD_RW = 0;
41     delay_ms(20);
42     LCD_DATA_PORT = 0x00;
43
44     LCD_sendCommand(LCD_TWO_LINES_EIGHT_BITS_MODE);
45     LCD_sendCommand(LCD_CURSOR_OFF);
46     LCD_sendCommand(LCD_CLEAR_COMMAND);
47 }
48
49 void LCD_sendCommand(unsigned char command) {
50     LCD_RS = 0; // Command mode
51     LCD_RW = 0; // Write mode
52     delay_ms(1);
53     LCD_E = 1;
54     delay_ms(1);
55     LCD_DATA_PORT = command;
56     delay_ms(1);
57     LCD_E = 0;
58     delay_ms(1);
59 }
60
61 void LCD_writeChar(unsigned char character) {
62     LCD_RS = 1; // Data mode
63     LCD_RW = 0; // Write mode
64     delay_ms(1);
65     LCD_E = 1;
66     delay_ms(1);
67     LCD_DATA_PORT = character;
68     delay_ms(1);
69     LCD_E = 0;
70     delay_ms(1);
71 }
72
73 void LCD_writeText(const char *text) {
74     while (*text) {
75         LCD_writeChar(*text++);
76     }
77 }
78
79 void LCD_clear(void) {
80     LCD_sendCommand(LCD_CLEAR_COMMAND);
81 }
82
83 void LCD_setCursor(unsigned char row, unsigned char col) {
84     unsigned char memory_address;
85
86     switch (row) {
87         case 0: memory_address = col; break;
88         case 1: memory_address = col + 0x40; break;
89         default: memory_address = col; break;
90     }
91     LCD_sendCommand(memory_address | LCD_SET_CURSOR_LOCATION);
92 }
```

• Keypad code

```
1 #ifndef KEYPAD_H
2 #define KEYPAD_H
3
4 #include <reg51.h>
5 #include "lcd.h"
6 #include "delay.h"
7
8 /* Define keypad configuration */
9 #define KEYPAD_NUM_COLS 3
10 #define KEYPAD_NUM_ROWS 4
11
12 /* Define ports for rows and columns */
13 #define KEYPAD_PORT P2
14
15 /* Button states */
16 #define KEYPAD_BUTTON_PRESSED 0
17 #define KEYPAD_BUTTON_RELEASED 1
18
```




```

18  /* Define pin mappings for rows */
19  sbit KEYPAD_ROW1 = KEYPAD_PORT^3;
20  sbit KEYPAD_ROW2 = KEYPAD_PORT^4;
21  sbit KEYPAD_ROW3 = KEYPAD_PORT^5;
22  sbit KEYPAD_ROW4 = KEYPAD_PORT^6;
23
24  /* Define pin mappings for columns */
25  sbit KEYPAD_COL1 = KEYPAD_PORT^0;
26  sbit KEYPAD_COL2 = KEYPAD_PORT^1;
27  sbit KEYPAD_COL3 = KEYPAD_PORT^2;
28
29  /* Function prototype */
30  unsigned char Keypad_getKey(void);
31
32  #endif /* KEYPAD_H */
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

➤ Microcontroller 2: Inside Home Unit

• Main code

```

1 #include <reg51.h>
2 #include "lcd.h"
3 #include "delay.h"
4 #include "button.h"
5 #include "leds.h"
6 #include "i2c_slave.h"
7
8 // Global variables
9 unsigned char AC_button_count = 0;
10 unsigned char Light_button_count = 0;
11 bit smart_mode = 0;
12
13 // Function prototypes
14 void LCD_showSmartFeatures(void);
15 void LCD_showHomeScreen(void);
16
17 void main(void) {
18     LCD_init();
19     I2C_Slave_Init();
20
21     LED_setWhite(LED_OFF);
22     LED_setBlue(LED_OFF);
23     LED_setYellow(LED_OFF);
24     LED_setOrange(LED_OFF);
25
26     LCD_showHomeScreen();
27
28     while (1) {
29         I2C_Slave_Check(); // Check for I2C 'S'
30
31         if (smart_mode) {
32             // AC Button Handling
33             if (Button_isACPressed() == BUTTON_PRESSED) {
34                 AC_button_count++;
35                 if (AC_button_count == 1) {
36                     LCD_moveCursor(0, 10);
37                     LCD_displayString("ON ");
38                 } else if (AC_button_count == 2) {
39                     AC_button_count = 0;
40                     LCD_moveCursor(0, 10);
41                     LCD_displayString("OFF");
42                 }
43             }
44         }
45     }
46 }

```



```

45 // Light Button Handling
46 if (Button_isLightPressed() == BUTTON_PRESSED)
47     Light_button_count++;
48     if (Light_button_count == 1) {
49         LCD_moveCursor(1, 14);
50         LCD_displayString("M1");
51         LED_setOrange(LED_ON);
52         LED_setYellow(LED_OFF);
53         LED_setBlue(LED_OFF);
54         LED_setWhite(LED_OFF);
55     } else if (Light_button_count == 2) {
56         LCD_moveCursor(1, 14);
57         LCD_displayString("M2");
58         LED_setOrange(LED_ON);
59         LED_setYellow(LED_ON);
60         LED_setBlue(LED_OFF);
61         LED_setWhite(LED_OFF);
62     } else if (Light_button_count == 3) {
63         LCD_moveCursor(1, 14);
64         LCD_displayString("M3");
65         LED_setOrange(LED_ON);
66         LED_setYellow(LED_ON);
67         LED_setBlue(LED_ON);
68         LED_setWhite(LED_OFF);
69     } else if (Light_button_count == 4) {
70         LCD_moveCursor(1, 14);
71         LCD_displayString("M4");
72         LED_setOrange(LED_ON);
73         LED_setYellow(LED_ON);
74         LED_setBlue(LED_ON);
75         LED_setWhite(LED_ON);
76     } else if (Light_button_count == 5) {
77         Light_button_count = 0;
78         LCD_moveCursor(1, 13);
79         LCD_displayString("OFF");
80         LED_setOrange(LED_OFF);
81         LED_setYellow(LED_OFF);
82         LED_setBlue(LED_OFF);
83         LED_setWhite(LED_OFF);
84     }
85 }
86 }
87 }
88 }
89
90 void LCD_showSmartFeatures(void) {
91     LCD_clearScreen();
92     LCD_moveCursor(0, 0);
93     LCD_displayString("AC Status: ");
94     LCD_moveCursor(1, 0);
95     LCD_displayString("Light Status:");
96 }
97
98 void LCD_showHomeScreen(void) {
99     LCD_clearScreen();
100     LCD_displayString(" Smart Home ");
101     LCD_moveCursor(1, 0);
102     LCD_displayString(" We Always Here ");
103 }
104

```

• LCD code

```

1 #ifndef LCD_H
2 #define LCD_H
3
4 #include <reg51.h>
5 #include "delay.h"
6
7 // LCD Control Pins
8 sbit LCD_RS = P0^1;
9 sbit LCD_En = P0^0;
10
11 // LCD Data Port
12 #define LCD_DATA_PORT P1
13
14 // LCD Command Constants
15 #define LCD_CLEAR_COMMAND 0x01
16 #define LCD_GO_TO_HOME 0x02
17 #define LCD_TWO_LINES_EIGHT_BITS_MODE 0x38
18 #define LCD_CURSOR_OFF 0x0C
19 #define LCD_CURSOR_ON 0x0E
20 #define LCD_SET_CURSOR_LOCATION 0x80
21
22 // LCD Function Prototypes
23 void LCD_init(void);
24 void LCD_sendCommand(unsigned char command);
25 void LCD_displayCharacter(unsigned char my_data);
26 void LCD_displayString(const char *Str);
27 void LCD_clearScreen(void);
28 void LCD_moveCursor(unsigned char row, unsigned char col);
29
30 #endif /* LCD_H */
31
32 #include <reg51.h>
33 #include "delay.h"
34 #include "lcd.h"
35
36 void LCD_init(void) {
37     LCD_RS = 0;
38     LCD_E = 0;
39     LCD_RW = 0;
40     delay_ms(20);
41     LCD_DATA_PORT = 0x00;
42 }

```



```

12 LCD_sendCommand(LCD_TWO_LINES_EIGHT_BITS_MODE);
13 LCD_sendCommand(LCD_CURSOR_OFF);
14 LCD_sendCommand(LCD_CLEAR_COMMAND);
15 }
16
17 void LCD_sendCommand(unsigned char command) {
18     LCD_RS = 0;    // Command mode
19     LCD_RW = 0;    // Write mode
20     delay_ms(1);
21     LCD_E = 1;
22     delay_ms(1);
23     LCD_DATA_PORT = command;
24     delay_ms(1);
25     LCD_E = 0;
26     delay_ms(1);
27 }
28
29 void LCD_writeChar(unsigned char character) {
30     LCD_RS = 1;    // Data mode
31     LCD_RW = 0;    // Write mode
32     delay_ms(1);
33     LCD_E = 1;
34     delay_ms(1);
35     LCD_DATA_PORT = character;
36     delay_ms(1);
37     LCD_E = 0;
38     delay_ms(1);
39 }
40
41 void LCD_writeText(const char *text) {
42     while (*text) {
43         LCD_writeChar(*text++);
44     }
45 }
46
47 void LCD_clear(void) {
48     LCD_sendCommand(LCD_CLEAR_COMMAND);
49 }
50
51 void LCD_setCursor(unsigned char row, unsigned char col) {
52     unsigned char memory_address;
53
54     switch (row) {
55         case 0: memory_address = col; break;
56         case 1: memory_address = col + 0x40; break;
57         default: memory_address = col; break;
58     }
59     LCD_sendCommand(memory_address | LCD_SET_CURSOR_LOCATION);
60 }
61

```

• LEDS code

```

1 #ifndef LEDS_H
2 #define LEDS_H
3
4 #include <reg51.h>
5
6 // LED Pin Definitions
7 sbit LED_WHITE = P0^5;
8 sbit LED_BLUE = P0^4;
9 sbit LED_YELLOW = P0^3;
10 sbit LED_ORANGE = P0^2;
11
12 // LED States
13 #define LED_ON 0
14 #define LED_OFF 1
15
16 // Function Prototypes
17 void LED_setWhite(unsigned char status);
18 void LED_setBlue(unsigned char status);
19 void LED_setYellow(unsigned char status);
20 void LED_setOrange(unsigned char status);
21
22 #endif /* LEDS_H */
23
24 #include "leds.h"
25
26 void LED_setWhite(unsigned char status) {
27     LED_WHITE = status;
28 }
29
30 void LED_setBlue(unsigned char status) {
31     LED_BLUE = status;
32 }
33
34 void LED_setYellow(unsigned char status) {
35     LED_YELLOW = status;
36 }
37
38 void LED_setOrange(unsigned char status) {
39     LED_ORANGE = status;
40 }
41

```



• BUTTON code

```
1 #include "button.h"
2 #include "delay.h"
3
4 unsigned char Button_isACPressed(void) {
5     if (AC_button == BUTTON_PRESSED) {
6         delay_ms(20); // debounce delay
7         if (AC_button == BUTTON_PRESSED) {
8             while (AC_button == BUTTON_PRESSED); //
9             return BUTTON_PRESSED;
10        }
11    }
12    return BUTTON_RELEASED;
13 }
14
15 unsigned char Button_isLightPressed(void) {
16     if (Light_button == BUTTON_PRESSED) {
17         delay_ms(20); // debounce delay
18         if (Light_button == BUTTON_PRESSED) {
19             while (Light_button == BUTTON_PRESSED);
20             return BUTTON_PRESSED;
21        }
22    }
23    return BUTTON_RELEASED;
24 }
25
26 #ifndef BUTTON_H
27 #define BUTTON_H
28
29 #include <reg51.h>
30
31 // Define Button Pins
32 sbit AC_button = P3^2;
33 sbit Light_button = P3^3;
34
35 // Define Button States
36 #define BUTTON_PRESSED 0
37 #define BUTTON_RELEASED 1
38
39 // Function Prototypes
40 unsigned char Button_isACPressed(void);
41 unsigned char Button_isLightPressed(void);
42
43 #endif /* BUTTON_H */
```

• I²C code

```
1 #ifndef I2C_SLAVE_H
2 #define I2C_SLAVE_H
3
4 void I2C_Slave_Init(void);
5 void I2C_Slave_Check(void);
6
7 #endif /* I2C_SLAVE_H */
8
9 #include <reg51.h>
10 #include "i2c_slave.h"
11
12 sbit SDA = P0^6;
13 sbit SCL = P0^7;
14
15 extern bit smart_mode;
16 void LCD_showSmartFeatures(void);
17
18 void I2C_Slave_Init(void) {
19     SDA = 1;
20     SCL = 1;
21 }
22
23 void I2C_Slave_Check(void) {
24     if (!SCL && !SDA) {
25         smart_mode = 1;
26         LCD_showSmartFeatures();
27     }
28 }
```

