University of Essex
CE323 Assignment

Advanced Embedded Systems Design

Home Alarm System

Name: Yu Wenlu
Essex ID: 1909135
wy19403@essex.ac.uk

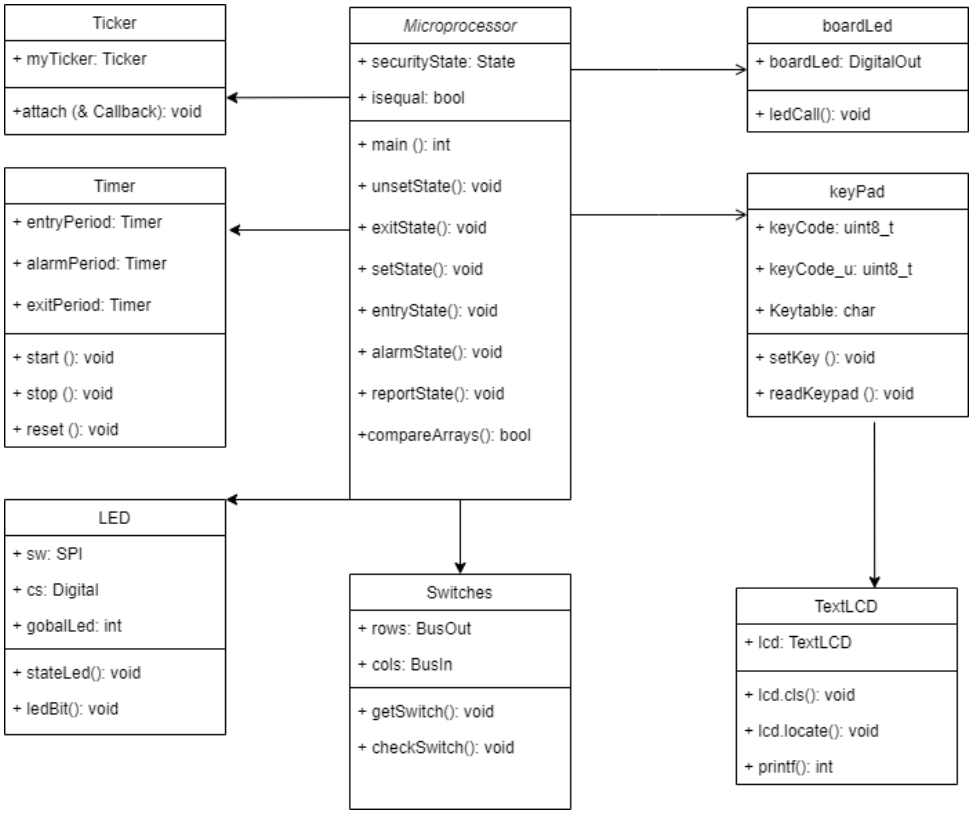
CONTENT

Requirement Form 2
Class Diagram 2
State Diagram 3
Sequence Diagram 4
Reference 5

Requirement Form

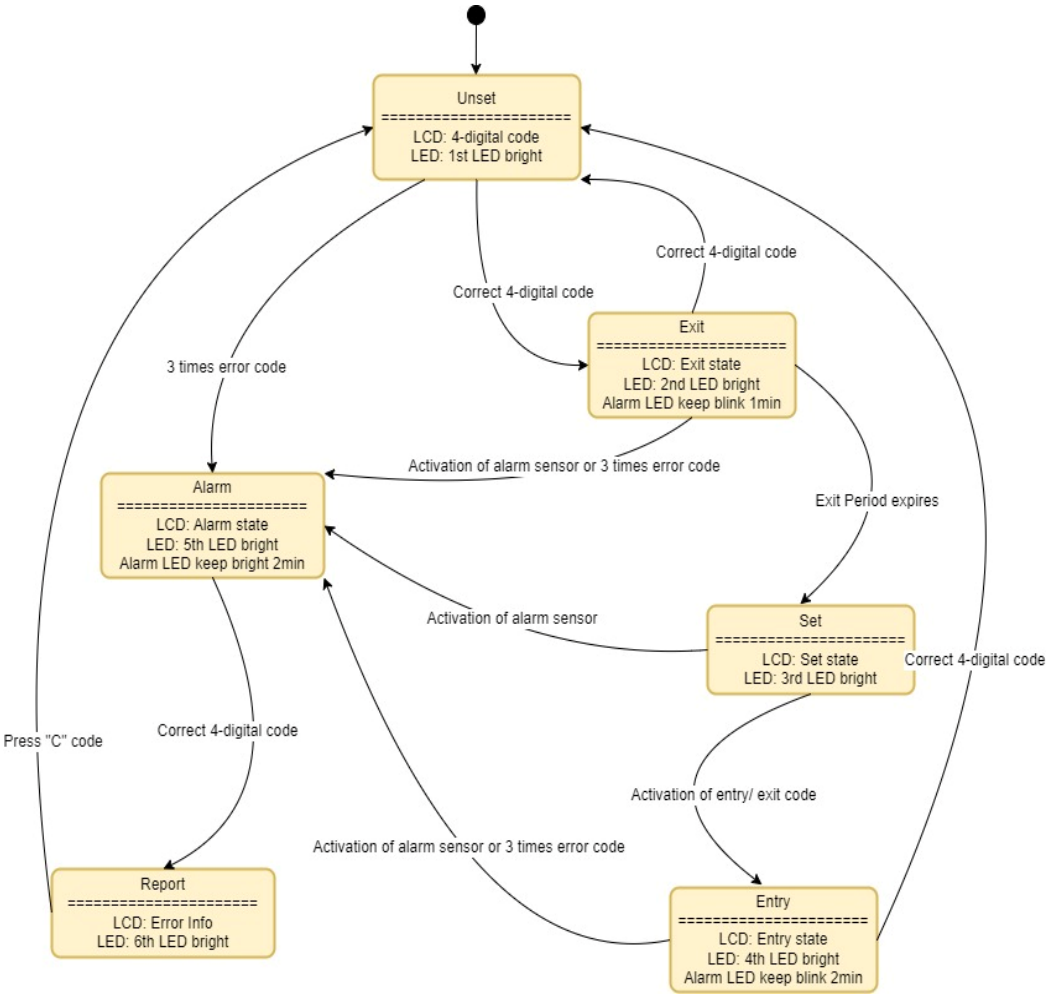
Name	Home Alarm System
Purpose	Sense intruders and send out alarm information
Inputs	Board's keyboard,8 switches
Outputs	8 * LED, 1* LCD screen, 1 * alarm-LED
Functions	The system uses a TextLCD display, a keypad, and LEDs to indicate the current state of the system, which can be one of six possible states: UNSET, EXIT, SET, ENTRY, ALARM, or REPORT.
Performance	The program uses a state machine to handle the transitions between states based on user input and sensor input. The state machine is implemented as an enum with six possible values. There are also several timer and ticker objects to manage the timing of various actions in the system.
Manufacture costs	£ 75.89
Physical size/weight	No more than 5 x 3.5 ich , 200 g
Power	221.1 mV

Class Diagram



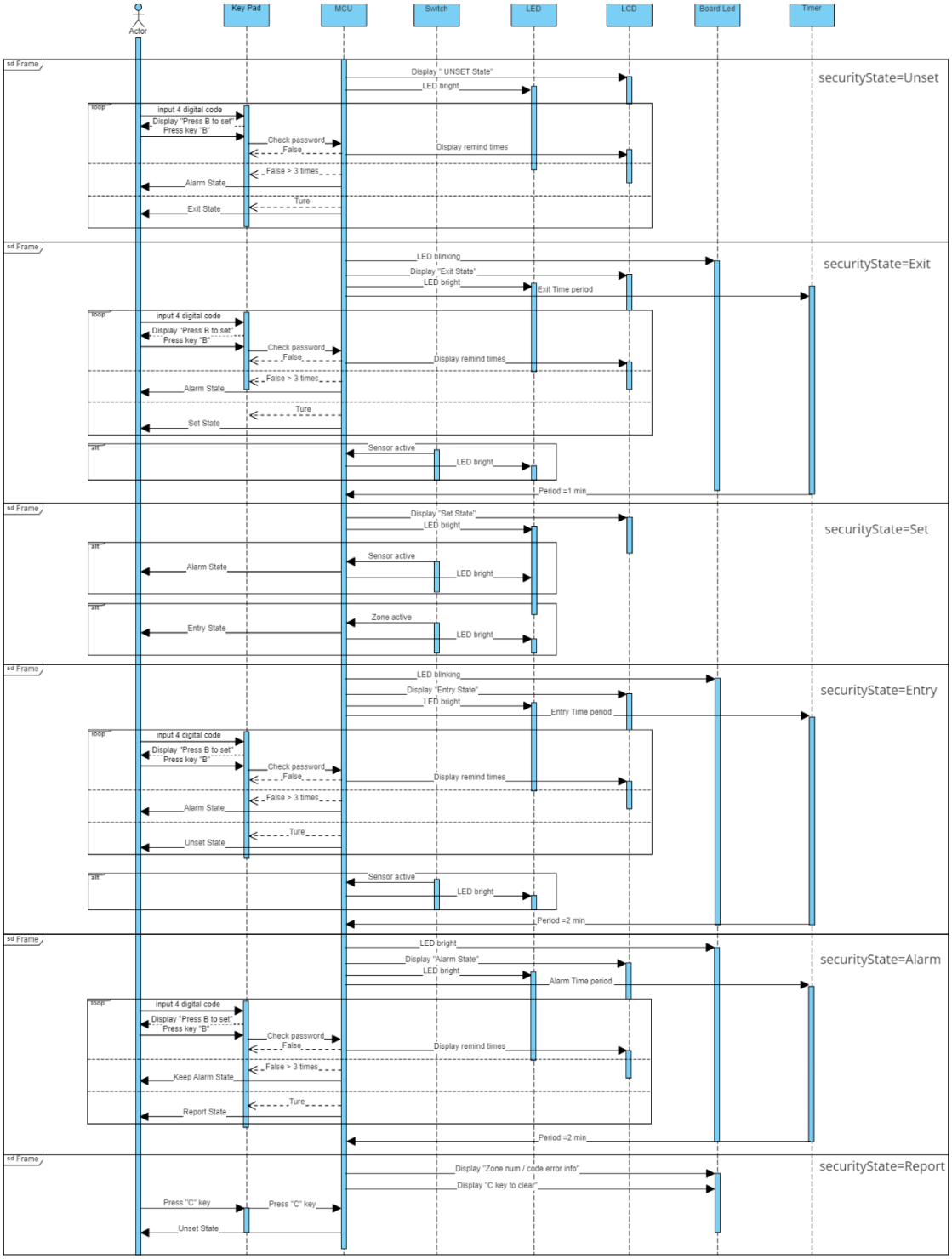
A class diagram is a type of diagram that shows the structure of a system by using classes, attributes, operations and relationships.

State Diagram



A state machine diagram is a type of diagram that shows the behavior of a system by using states, transitions and events [1]. A state is a condition or situation that an object can be in [2]. A transition is a change from one state to another triggered by an event [3]. An event is something that happens and affects the system [3].

Sequence Diagram



Reference

[1] V. Paradigm., “What is State Machine Diagram?,” [online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/. [Access data: 24 3 2023].

[2] Lucidchart., “State Machine Diagram Tutorial,” [online]. Available: https://www.lucidchart.com/pages/uml-state-machine-diagram . [Access data: 23 3 2023].

[3] Wikipedia, “UML state machine,” [online]. Available: https://en.wikipedia.org/wiki/UML_state_machine. [Access data: 24 3 2023].

Appendix

```
/* @Copyright University of Essex 2017
 * @Author: NicoleYu wy19403@essex.ac.uk
 * @Date: 2023-03-03 22:20:33
 * @LastEditors: NicoleYu wy19403@essex.ac.uk
 * @LastEditTime: 2023-03-22 16:11:55
 * @FilePath: \mbed-os-example-blinky\assignment.cpp
 * @FILE - main.c
        mbed LPC1768 application.
 * @DESCRIPTION
        hardware: mbed LPC1768, an extension board.
        Use a terminal program such as 'HyperTerminal' to communicate with the application.
        The serial port configuration is shown below.
            9600 Baud
            8 data bits
            No parity bits
            One stop bit
            No hardware flow control
============================================================*/

#include "mbed.h"
#include "TextLCD.h"

//---------Board initialization declaration----------
TextLCD lcd(p15, p16, p17, p18, p19, p20); // Create a TextLCD object with pins for rs, e, d4-d7
BusOut rows(p26,p25,p24);// Create a TextLCD object with pins for rs, e, d4-d7
BusIn cols(p14,p13,p12,p11);// Create a BusIn object with pins for columns
SPI sw(p5, p6, p7);// Create an SPI object with pins for mosi, miso and sclk
DigitalOut cs(p8); // Create a DigitalOut object with pin for cs
DigitalOut boardLed(LED1);// Create a DigitalOut object with pin for boardLed

Ticker myTicker;// Create Ticker objects for light, monitorZones, blink, blinkk and bright
Timer exitPeriod, entryPeriod, alarmPeriod;// Create Timer objects for exitPeriod, entryPeriod and alarmPeriod

char Keytable[] = { 'F', 'E', 'D', 'C',
                    '3', '6', '9', 'B',
                    '2', '5', '8', '0',
                    '1', '4', '7', 'A'
                  };

```

```
40. //----------State declaration------------------------
41. enum State {UNSET, EXIT, SET, ENTRY, ALARM, REPORT} securityState; // State machine
    setting
42. const char* StateNames[] = {"UNSET", "EXIT", "SET", "ENTRY", "ALARM", "REPORT"}; //
    In order to display the current status on the lcd screen conveniently
43.
44. //----------Personalization declaration-----------------
45. uint8_t keyCode[4] = {'1','2','3','4'}; // Correct password
46. uint8_t keyCode_u[4] = {'1','2','3','4'}; // User input password
47.
48. //----------Intermediate variable declaration----------------
49. int judge=0;//'judge' variable used to judge the state. 0=unset, 1=other, 2=unset,
    3=alarm.
50. int gobalLed = 0;// gobalLed is ALARMLED on the development board LED
51. int counter = 0; // counter is 4-digit code attempt times, and can not more than 3.
52.
53. //----------Function declaration----------------
54. void initLeds();
55. char getKey();
56. int getSwitch();
57. void readKeypad();
58. bool compareArrays(const uint8_t* keyCode, const uint8_t* keyCode_u);
59. void ledBit(int order);
60. void checkSwitches();
61. void setKey(void);
62. void stateLed();
63. void ledCall();
64. void unsetState();
65. void exitState();
66. void setState();
67. void entryState();
68. void alarmState();
69. void reportState();
70. int main();
71.
72. //------------------Code begining------------------------
73.
74. // Initialize the LED
75. void initLeds() {
76.
77.     cs = 0;// latch must start low
78.     sw.format(16,0);// SPI 16 bit data, low state, high going clock
79.     sw.frequency(1000000);// 1MHz clock rate
80.
```

```
}

char getKey(){
    int i,j;
    char ch=' ';

    for (i = 0; i <= 3; i++) {
        rows = i;
        for (j = 0; j <= 3; j++) {
            if (((cols ^ 0x00FF)  & (0x0001<<j)) != 0) {
                ch = Keytable[(i * 4) + j];
            }
        }
    }
    return ch;
}
void readKeypad(){
    char b = ' ';
    b=getKey();
          wait_ns(10000);
          if (b != ' '){
              setKey();
          }
      }

      // Verify that the password is correct
      bool compareArrays(const uint8_t* keyCode, const uint8_t* keyCode_u) {
          for (int i = 0; i < 3; i++) {
              if (keyCode[i] != keyCode_u[i]) {
                  return 0;
              }
          }
          return 1;
      }
      //controlling the state of one or more LEDs using a shift register.
      void ledBit(int order) {
          order = (((8-order) & 0x0007) + 1) * 2;
   //offset of led state in 'gobalLed'
       gobalLed = gobalLed & ((0x0003 << order) ^ 0xffff); // clear and set led sta
   te
          sw.write((gobalLed & 0x03ff) | ((gobalLed & 0xa800) >> 1) | ((gobalLed &
   0x5400) << 1));
          cs = 1;                                    // latch pulse start
          cs = 0;
```

```
122.      }
123.
124.      void checkSwitches(){
125.          initLeds();
126.          int switches = getSwitch();
127.          if ((securityState != UNSET)&& (securityState != ALARM) && (securityState
      != REPORT)&&(num != 0x80)&&(num !=0x00)){
128.
129.              judge = 1;
130.              counter = 0;
131.              securityState = ALARM;
132.              exitPeriod.stop();
133.              exitPeriod.reset();
134.              lcd.cls();
135.              alarmState();
136.          }
137.      }
138.
139.      int getSwitch()
140.      {
141.          int switches = cols;
142.          switches = switches*17
143.          for(int i=0;i<=7;i++){
144.              if ((switches & 0x0001<<i)!=0){
145.                  gobalLed  = gobalLed | (0x0003 << i*2); }
146.              else {
147.                  gobalLed  = gobalLed & ((0x0003 << i*2) ^ 0xffff); }
148.          }
149.          sw.write(gobalLed);
150.          cs = 1;
151.          cs = 0;
152.          return switches;
153.          ///([1])
154.      }
155.      void setKey(void){
156.          int a;
157.          char b = ' ';
158.          judge =0;
159.          b=getKey();
160.          wait_ns(10000);
161.          lcd.cls();
162.          lcd.locate(0,0);
163.          lcd.printf("STATE: %s", StateNames[securityState]);
164.          lcd.locate(0,1);
```

```
165.          lcd.printf("Code:____");
166.          for(a=0;a<4;a++){
167.              b=getKey();
168.              wait_ns(10000);
169.              if (judge != 0){
170.                  judge = 0;
171.                  lcd.cls();
172.                  return;
173.              }
174.              switch(b){
175.                  case ' ':
176.                      a--;
177.                      break;
178.                  case 'C':
179.                      if(a>0){
180.                          a=a-2;
181.                          lcd.locate(6+a,1);
182.                          lcd.putc('_');
183.                      }
184.                      else if(a==0){
185.                          a--;
186.                      }
187.                      break;
188.                  case 'B':
189.                      default:
190.                      lcd.locate(5+a,1);
191.                      lcd.putc('*');
192.                      keyCode_u[a]=b;
193.                      break;
194.              }    // [3]
195.          }
196.          b=getKey();
197.          wait_ns(10000);
198.          while (b != 'B'){
199.
200.              if (judge != 0){
201.                  judge = 0;
202.                  lcd.cls();
203.                  return;
204.              }
205.              lcd.locate(0,1);
206.              lcd.printf("Press B to set");
207.              b=getKey();
208.              wait_ns(100);
```

```
209.          }
210.          if(bool isEqual = compareArrays(keyCode, keyCode_u)){
211.              if(securityState == UNSET){
212.                  securityState = EXIT;
213.                  lcd.cls();
214.              }
215.              else if(securityState == EXIT){
216.                  securityState = UNSET;
217.                  exitPeriod.stop();
218.                  exitPeriod.reset();
219.                  boardLed = 0;
220.                  lcd.cls();
221.              }
222.              else if(securityState == ENTRY){
223.                  securityState = UNSET;
224.                  entryPeriod.stop();
225.                  entryPeriod.reset();
226.                  boardLed = 0;
227.                  lcd.cls();
228.              }
229.              else if(securityState == ALARM){
230.                  securityState = REPORT;
231.                  alarmPeriod.stop();
232.                  alarmPeriod.reset();
233.                  lcd.cls();
234.                  reportState();
235.              }
236.              counter = 0;
237.          }
238.          else {
239.              counter = counter++;
240.              if(counter >= 3){
241.                  counter = 0;
242.                  securityState = ALARM;
243.                  lcd.cls();
244.                  alarmState();
245.              }
246.              else{
247.                  lcd.locate(0,1);
248.                  lcd.printf("Remain attempt:%d", 3-counter);
249.                  wait_ns(10000);
250.              }
251.          }
252.      }
```

```
253.      void stateLed(){
254.          switch(securityState)
255.              {
256.                  case UNSET :
257.                      ledBit(1);
258.                      break;
259.                  case EXIT :
260.                      ledBit(2);
261.                      break;
262.                  case SET :
263.                      ledBit(3);
264.                      break;
265.                  case ENTRY :
266.                      ledBit(4);
267.                      break;
268.                  case ALARM :
269.                      ledBit(5);
270.                      break;
271.                   case REPORT :
272.                      ledBit(6);
273.                      break;
274.              }
275.      }
276.      //controlling the sort of security system with LED lights and an LCD screen.
277.      void ledCall(){
278.          if ((exitPeriod.read()<=10) && (securityState == EXIT) &&(securityState =
    = ENTRY)){
279.              boardLed = !boardLed;
280.          }
281.
282.          else if((exitPeriod.read()>10)){
283.
284.              exitPeriod.stop();
285.              exitPeriod.reset();
286.              counter = 0;
287.
288.              if (securityState == EXIT){
289.                  judge = 2;
290.                  securityState = SET;
291.                  boardLed = 0;
292.                  lcd.cls();
293.                  setState();
294.              }
295.              else if(securityState == ENTRY){
```

```
296.                 judge = 3;
297.                 securityState = ALARM;
298.                 lcd.cls();
299.                 alarmState();
300.             }
301.
302.         }
303.         else if (securityState == ALARM){
304.             alarmPeriod.start();
305.             boardLed = 1;
306.             if(alarmPeriod.read()>20){
307.                 boardLed = 0;
308.                 alarmPeriod.stop();
309.                 alarmPeriod.reset();
310.             }
311.         }
312.     }
313.     // unset state function
314.     void unsetState(){
315.         lcd.locate(0,0);
316.         lcd.printf("STATE: UNSET");
317.         ledBit(1);
318.         readKeypad();
319.     }
320.     // exit state function
321.     void exitState(){
322.         exitPeriod.start();
323.         lcd.locate(0,0);
324.         lcd.printf("STATE: EXIT");
325.         ledBit(2);
326.         readKeypad();
327.     }
328.     // set state function
329.     void setState(){
330.         lcd.locate(0,0);
331.         lcd.printf("STATE: SET");
332.         ledBit(3);
333.         if(num == 0x80){
334.             counter = 0;
335.             securityState = ENTRY;
336.             lcd.cls();
337.         }
338.
339.     }
```

```
340.      // entry state function
341.      void entryState(){
342.          entryPeriod.start();
343.          lcd.locate(0,0);
344.          lcd.printf("STATE: ENTRY");
345.          ledBit(4);
346.          readKeypad();
347.      }
348.      //alarm state function
349.      void alarmState(){
350.
351.          alarmPeriod.start();
352.          lcd.locate(0,0);
353.          lcd.printf("STATE: ALARM");
354.          ledBit(5);
355.          readKeypad();
356.      }
357.      //report state function
358.      void reportState(){
359.          char cls=' ';
360.          if ( (checkSwitches() & 0x03) != 0 )
361.               lcd.printf("full set error");
362.
363.       if ( (checkSwitches() & 0x0c) != 0 )
364.                lcd.printf("set error");
365.
366.       if ( (checkSwitches() & 0x30) != 0 )
367.                lcd.printf("entry error");
368.
369.       if ( (checkSwitches() & 0xc0) != 0 )
370.                lcd.printf("exit error");
371.          else
372.                lcd.printf("code error");
373.
374.          lcd.locate(0,1);
375.          lcd.printf("C key to clear");
376.          cls = getKey();
377.          wait_ns(10000);
378.
379.          if (cls == 'C'){
380.              securityState = UNSET;
381.              lcd.cls();
382.          }
383.          ledBit(6);
```

```
384.      }
385.      int main() {
386.
387.          initLeds(); //initialize leds
388.          lcdRefresh(); // initial lcd clear
389.          securityState = UNSET; //initial state
390.          monitorZones.attach_us(&checkSwitches, 0.1); //check zones every 100ms
391.          myTicker.attach(&ledCall, 0.1);// check LED
392.          while(1)
393.          { // State machine judgment
394.              switch(securityState)
395.              {
396.                  case UNSET :
397.                      unsetState();
398.                      break;
399.                  case EXIT :
400.                      exitState();
401.                      break;
402.                  case SET :
403.                      setState();
404.                      break;
405.                  case ENTRY :
406.                      entryState();
407.                      break;
408.                  case ALARM :
409.                      alarmState();
410.                      break;
411.                  case REPORT :
412.                      reportState();
413.                      break;
414.                  //[4]
415.              }
416.          }
417.
418.      }
419.      // reference
420.      // [1]"home_alarm_simple - | Mbed," mbed import http://os.mbed.com/users/liru
    ihao2008/code/home_alarm_simple/.
421.      // [2]"CE323_Lab4"
422.      // [3]"CE323_Lab6"
423.      // [4]"CE323_Lab7"
```