Module Name: CE323

Student Name: OKIERETE EDU

Student ID: 1904971

HOME ALARM SYSTEM

TABLE OF CONTENTS

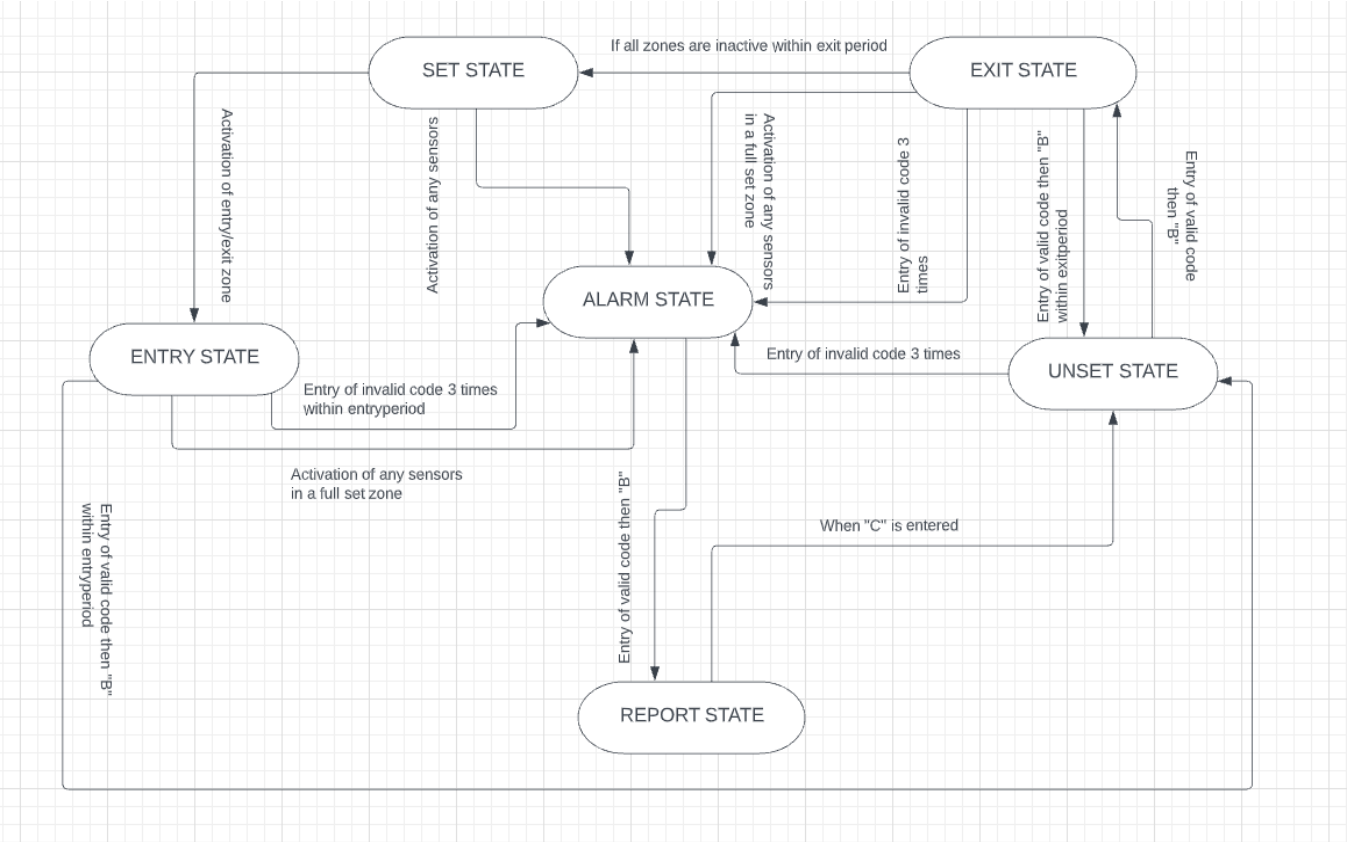
Abstract

A home Alarm System, is a device with sensors and a sounding alarm when a possible entrance in a home has been breached. When the sensors such as the movement sensors, pressure mats and magnetic contacts are triggered or the code Keyed into the keypad which lets the user into the home is wrong a certain number of times, the Alarm state will be notified and this should cause a sound. These sensors and code communicate with a control panel to secure the home.

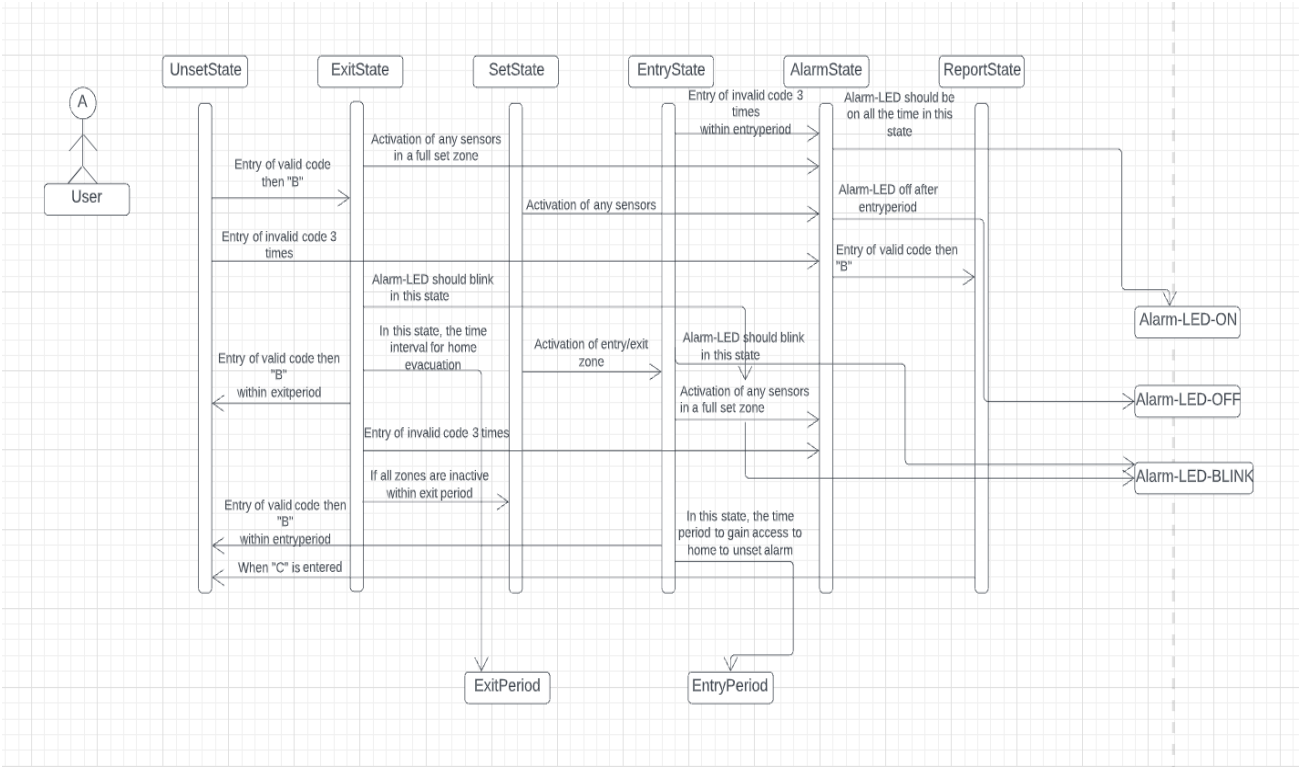
Introduction

The ARM mbed NXP LPC1768 microcontroller will be used alongside mbed compiler to be able to create a Home Alarm System. In this system, there is a correct Keycode used to get into the house and if this Keycode is wrong three times, the alarm-LED on the microcontroller will turn on until the user enters the correct code. This system uses the idea of a finite state machine and switches between states. There are 6 states which will be go back and forth with each other depending on the input in the Keypad or the signal from the sensors.

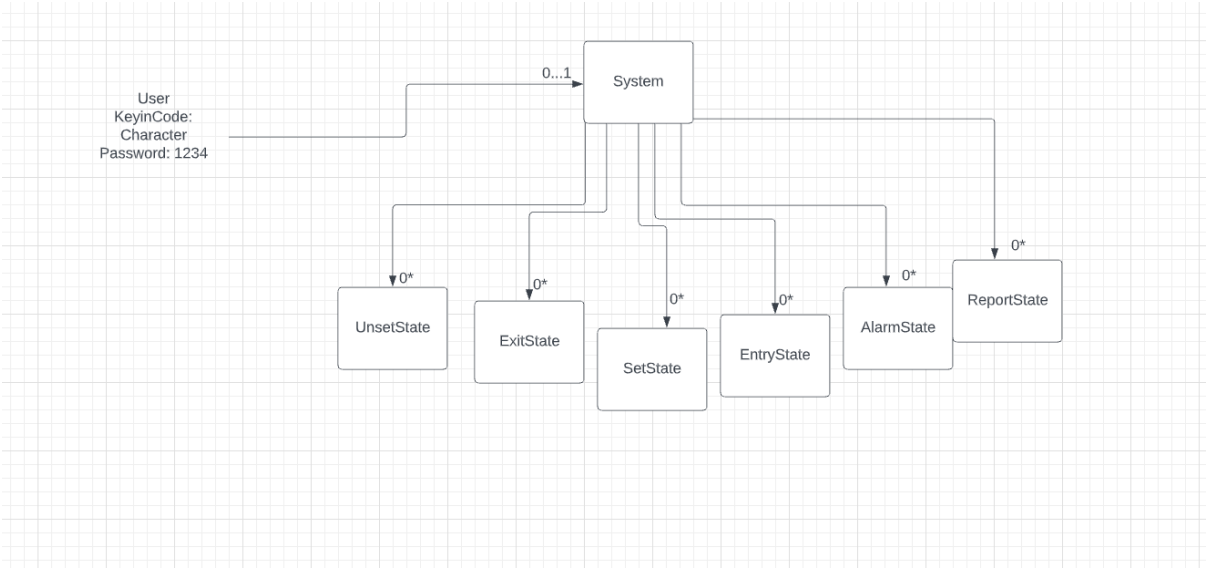
State Machine Diagram



Sequence Diagram



CLASS DIAGRAM



APPENDIX

```
#include "mbed.h"
//#include "main.h"
#include "TextLCD.h"

TextLCD lcd(p15, p16, p17, p18, p19, p20);
DigitalOut Alarmled(LED1);
BusOut rows(p26, p25, p24);
BusIn cols(p14, p13, p13, p11);
Timeout flipper;
Timeout flipper1;

SPI sw(p5, p7, p6);
DigitalOut cs(p8);
//--------------------------------------------------------------------------
---//

int MoveSensor = 0;

int PressureMats = 0;

int MagneticContact = 0;

int report_Var = 0;


char input[] = {'_','_','_','_'}; // character array to store what the user
inputs

const char KeyinCode [] = {'1', '2', '3', '4'}; //character array to store
the keyCode, alarm coode combination

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

char getKey () // getting the keys from the keypad
{
     int i, j; // i to store row of keypad and j to store the columns of
the keypad
     char let = ' ';

     for (i = 0; i <= 3; i++){  //row increment
        rows = i;
        for (j =0; j<=3; j++){ //column increment
             if(((cols ^ 0x00FF) & (0x0001<<j)) != 0) {
                 let = Keytable[(i  *  4)+ j];
            }
            }
            }
    return let;
}

void setKey()
{
   int r;
   char s;
```

```
   lcd.locate(0,0);
   lcd.printf("UnsetState");
   lcd.locate(0,1);
   lcd.printf("Enter Code: ____"); //when the user types in the wrong code,
they are sent to an unset state to retype this code. After the wrong try
the third time it will go to alarm state
   for(r = 0; r < 4; r++){ //for loop for 4 digit code input
        s = getKey();
        wait(0.2);  // setting a delay would allow to get charcaters one at
a time
        switch(s){
            case ' ': // space for getting four of the code numbers
              r--;
             break;
            case 'C': //if the user presses the "C" key, then the last
entered character should be deleted and replaced with "_"
              if(r>0){
                  r = r-2; //deleting the previous input number if r>0
                  lcd.locate(9+r, 1);
                  lcd.putc('_');
                  }
                else if (r==0){
                    r--; // if r==0 there are no more input hence back to
the begining
                    }
                 break;
            default:

                case 'B':
                     if (s != 'B'){        // if 'B' is not pressed
                         lcd.locate(8+r, 1);
                         lcd.putc('*');   //input being replaced by "*"
with each successive digit entered.
                         input[r] = s;
                         }
                        else if (s == 'B'){
                            r--;
                         }
         }
    }
     wait(1);

    int main(); {
        char s;
        int i;
        int chars_count = 0;
        lcd.cls();

        while (1){
            setKey();
            chars_count = 0;


            for (i = 0; i< 4; i++){
                    if(input[i] == KeyinCode[i]){    //seeing if each
input is same as Key in code
                        chars_count = chars_count + 1;  // increment the
array
                        if (chars_count > 4){           // when the
characters typed in are greater than 4, set back to 0
```

```
                                chars_count = 0;            // character count
= 0
                            }
                        }
                    }

    lcd.cls();
    lcd.locate(0,0);
    lcd.printf("Press B to set");
    s = getKey();

            while(s != 'B'){
                s = getKey();
                if (s == 'B'){ //if 'B' is pressed
                    if (chars_count == 4){  // if the character count is 4,
the input matches the KeyinCode
                        lcd.cls();
                        lcd.locate(0,0);
                        lcd.printf("ExitState");
                    }
                    else if (chars_count != 4){  // if the character count
is not 4, the input doesn't match the KeyinCode
                        lcd.cls();    // go back to setkey function follow
loop and try again
                        lcd.locate(0,0);
                     }

                }
            }


            }
        }

void AlarmState();{
    Alarmled = 1;// When in the alarm state, the alarm-LED should be on all
the time.
    wait(120);//After 2 minutes, the alarm-LED unit should be disabled.
    if(input[] == KeyinCode[]){
    Alarmled = 0;
    ReportState();//If the user enters the correct code followed by "B",
the alarm should change to the report state,

    }
      else if(input[] !== KeyinCode[]){
        AlarmState();
        }
    setKey();
    }

    void EntryState();{
    while(1){ //alarm led blinking as it's at entry state
        Alarmled = 1;// Whilst in the entry state, the alarm-LED should
sound blink
        wait(0.5);
        Alarmled = 0;
        wait(0.5);
        setKey();
      if(MoveSensor == 1 || PressureMats == 1 || MagneticContact == 1 ){//
        Alarmled = 1;
        }
```

```
      flipper.attach(&entryperiod, 120.0);//2 minutes is set time to unset
alarm, timer interrupt
      //if(MoveSensor == 0 && PressureMats == 0 && MagneticContact == 0){
       //flipper.attach();
       }

    }
    void entryexit(){//assigning zone sensor to a switch
      cs=0;
    sw.format(16,0);
    sw.frequency(1000000);
    while(1){

    rows = 4;
    int switches = cols;
    rows = 5;
    switches = switches*16 + cols;

    if(switches == 1){
        sw.write(0x0002);//it will light when at this
        cs = 1;
        cs = 0;
        wait(0.25);
        lcd.cls();
        lcd.locate(1,1);
        lcd.printf("%d", switches);
        EntryState();
        }
        else{
        sw.write(0x0000);
        cs = 1;
        cs = 0;
        }
    }
}


void UnsetState() {
    if(MoveSensor == 1 || PressureMats == 1 || MagneticContact == 1 ){
        Alarmled = 0; //In the unset state, activation of any of the
sensors should not cause the alarm-LED to blink.
        }

        setKey(); // setting key after getting key
        //while(setKey()){
            //Alarmled = 0; //alarmled off
            //}

          // pad();
           if(code == 1){
               ExitState();
               }

        }


void entryperiod();{ //is the time to gain access to the home system to
unset the alarm
    AlarmState();
    }
```

```
void EntryState();{
    while(1){ //alarm led blinking as it's at entry state
        Alarmled = 1;// Whilst in the entry state, the alarm-LED should
sound blink
        wait(0.5);
        Alarmled = 0;
        wait(0.5);
        setKey();
      if(MoveSensor == 1 || PressureMats == 1 || MagneticContact == 1 ){//
        Alarmled = 1;
        }
      flipper1.attach(&entryperiod, 120.0);//2 minutes is set time to unset
alarm, timer interrupt
      //if(MoveSensor == 0 && PressureMats == 0 && MagneticContact == 0){
       //flipper.attach();
       }

    }

void SetState();
{
    lcd.cls();
    lcd.locate(1,1);
    lcd.printf("Set State");



  if(MoveSensor == 1){
    report_Var = 1;
    }
  if(PressureMats == 1){
    report_Var = 2;
    }
  if(MagneticContact == 1){
    report_Var = 3;
    }
 if(MoveSensor == 1 || PressureMats == 1 || MagneticContact == 1 ){
       AlarmState(); //In the set state, activation of any sensors in the
set state zone should cause the system to enter the alarm state.
    }




    EntryState();  //{ Activation of the entry / exit zone should change
the state to entry state.
}

void exitperiod();{
    SetState();
    }

    void ExitState()
    {
        lcd.cls();
        lcd.locate(1,1);
        lcd.printf("Exit State");
        if(MoveSensor == 0 && PressureMats == 0 && MagneticContact == 0){
       flipper.attach(&exitperiod, 60.0); // If all the zones are inactive
when the exit period (e. g. 1 minutes) expires, the exit state should enter
the set state.
```

```
        }

         while(1){ //Whilst in the exit state the alarm-LED should blink
         Alarmled = 1;
         wait(1);
         Alarmled = 0;
         wait(1);
         if(MoveSensor == 1 || PressureMats == 1 || MagneticContact == 1 ){
        AlarmState(); //In the exit state, activation of any sensors in a
full set zone should cause the alarm to enter the alarm state.
         }
         setKey();
    }



    }


void ReportState();{
    lcd.cls();
    lcd.locate(0,0);



    switch(report_Var) {//showing zone numbers or code error information
  case 1:
    lcd.printf ("Movement Sensor");
    break;
  case 2:
    lcd.printf ("Pressure Sensor");
    break;
  case 3:
    lcd.printf ("Magnetic Sensor");
    break;

  default:
lcd.cls();
    lcd.locate(0,0);
    lcd.printf("Press C to Clear");
    s = getKey();

            while(s!= 'C'){                                                    //
making sure only 'B' is pressed to continue
                s = getKey();
// getting the key input
                if (s == 'C'){
                    if (chars_count == 4){
// if the character count is 4. i.e if  the code is correct

                     }

                  }
                }
}



    }
```

```
//-------------------------------------------------------------------------
---//

void MoveSensor(){//assigning each sensor to a switch
    cs=0;
    sw.format(16,0);
    sw.frequency(1000000);

 while(1){

    rows = 4; //accesssing pins and switches
    int switches = cols;
    rows = 5;
    switches = switches*16 + cols;

    if (switches == 128){
        sw.write(0x4000);
        cs = 1;
        cs = 0;
        wait(1);
       lcd.cls();
       lcd.locate(1,1);
       lcd.printf("%d", switches);
       }
       else{
        sw.write(0x0000);
        cs = 1;
        cs = 0;
        }
    }
}

void PressureMats(){
    cs=0;
    sw.format(16,0);
    sw.frequency(1000000);
    while(1){

    rows = 4;
    int switches = cols;
    rows = 5;
    switches = switches*16 + cols;

     if(switches == 32){
        sw.write(0x0400);
        cs = 1;
        cs = 0;
        wait(0.25);
       lcd.cls();
       lcd.locate(1,1);
       lcd.printf("%d", switches);
       }
       else{
        sw.write(0x0000);
        cs = 1;
        cs = 0;
        }
    }
}

void MagneticContact(){
```

```
    cs=0;
    sw.format(16,0);
    sw.frequency(1000000);
    while(1){

    rows = 4;
    int switches = cols;
    rows = 5;
    switches = switches*16 + cols;

    if(switches == 64){
        sw.write(0x1000);
        cs = 1;
        cs = 0;
        wait(0.25);
        lcd.cls();
        lcd.locate(1,1);
        lcd.printf("%d", switches);
        }
        else{
        sw.write(0x0000);
        cs = 1;
        cs = 0;
        }
    }
}
```

```
//-------------------------------------------------------------------------
---//
int main(){

  UnsetState();

    while(1){
        switch(securitystate)//to switch case
        {
            case UNSET:
            UnsetState();
            int count = 0;
            setKey();
            while(count < 2){
            if( input[i] == KeyinCode[i]){    //seeing if each input is
same as Key in code
                          chars_count = chars_count + 1;){
                          ExitState();
                         }
                  else {
                          count++;
                        setKey();
                         }
                  if(count == 2){
                   AlarmState();
                   }
                }

                break;
```

```
case EXIT:
ExitState();
int count = 0;
setKey();
while(count < 2){
if(input[i] == KeyinCode[i]){
             chars_count = chars_count + 1;){
             UnsetState();
            }
      else {
             count++;
            setKey();
            }

      if(count == 2){
       AlarmState();
       }
    }
      break;
case SET:
  SetState();
break;

case ENTRY:
  EntryState();
int count = 0;
setKey();
while(count < 2){
if(input[i] == KeyinCode[i]){
             chars_count = chars_count + 1;)
             UnsetState();

            }
      else {
             count++;
            setKey();
            }
      if(count == 2){
       AlarmState();
       }
      }
      break;

case ALARM:
AlarmState();
int count = 0;
setKey();
while(count < 2){
if(input[i] == KeyinCode[i]){
             chars_count = chars_count + 1;)
             ReportState();
            }
      else {
             count++;
            setKey();
            }

      if(count == 2){
       AlarmState();
      }
```

```
            }
        break;

        case REPORT:
        ReportState();
        break;
    }
}
}
```