Cairo University
Faculty of Engineering
Healthcare Engineering and Management
Credit Hours System

Smart Hospital Bed

12th December 2021

TEAM MEMBERS

1. Mohamed Nasser Mohamed **1190438**
2. Hassan Samy Mohamed **1190025**
3. Zeyad Mansour Kassem **1190368**

GITHUB LINK

The code was created and posted as a public repository on github

Link: Smart_Hospital_Bed_AVR: An automatic temperature & posture control hospital bed prototype for SBEN330 (github.com)

OVERVIEW

Adapting current technology into the field of hospital beds for both patient and physician convenience, making them both ergonomic and user-friendly and oriented is a very important yet lacking aspect of our current healthcare system. Hence, an array of sensors as well as other embedded electronics would be used to evolve the current mainstream barely-functional hospital beds into what we'd call a "smart" hospital bed.

FEATURES

Weight Tracking

Patients admitted long-term with a difficulty in normal intake of nutrition often lose significant weight during their stay. While glucose and other macro and micronutrient solutions help maintain their minimum nutrition requirements, they are often insufficient to maintain their body weight long-term. Constant automatic monitoring and detection of severe weight loss directly embedded into the hospital bed would prove to be very essential and convenient to both patient and physicians.

Occupancy Tracking

With hospital information systems becoming essential building blocks of any healthcare facility nowadays, tracking the availability of hospital beds for incoming patients is of utmost importance. A simple pressure/weight sensor would determine the occupancy of the beds, where this data would be fed through the information system.

Body Temperature Monitor

A simple infrared temperature sensor constantly monitoring the patient's body temperature, displaying it, and alarming the physician of any abnormal increase in body temperature would also be more convenient to both parties in cases where body temperature monitoring is necessary.

LCD User Interface

For the info collected and monitored by the smart bed to be readily available and displayed, an LCD display would be attached to existing monitor displays (such as those displaying essential body parameters) and would be constantly viewing these data.

Bed Heater

The bed contains a heater below the mattress, whenever the temperature goes below a threshold, refer to usage modes, no.3

USAGE MODES

1. Automatic Sleep Mode

A button is pressed by the nurse or patient to indicate that they are preparing to sleep this will result in the following series of actions:

a. Servo motor lays the headrest back
b. Light is dimmed (relay or dimmer)
c. LCD displays “Sleep mode activated + temperature”

2. Upright (Sitting)

This is the default mode

a. Servo motor is in a 50 degree position
b. Light is turned on
c. LCD displays “Patient is upright + temperature”’

3. Automatic Bed Heating

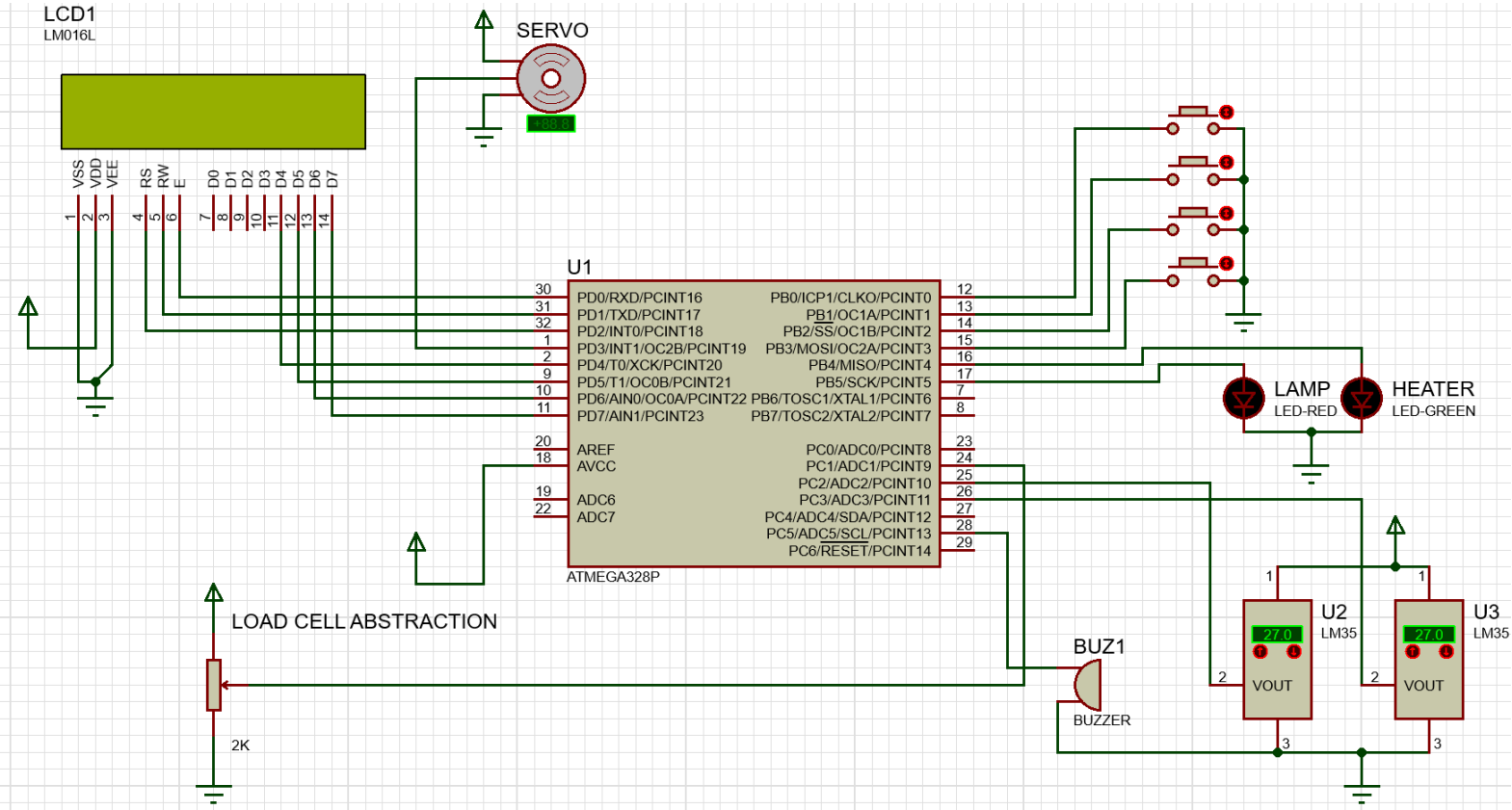
This enables the heater whenever the ambient temperature sensor goes below a preset threshold, this threshold will be set using up and down buttons in the interface

ALARMS

1. Normal body temperature exceeded
2. Maximum rated weight exceeded

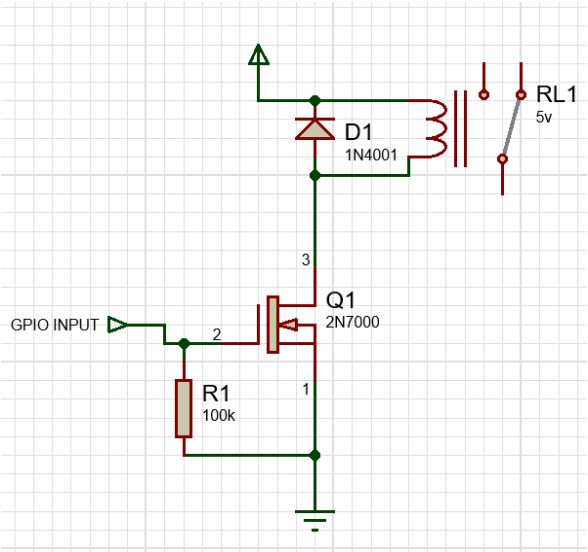
CIRCUIT DIAGRAM

Functional Circuit Diagram



This is a simplified circuit diagram demonstrating all port connections. (LEDs are just an abstraction of the relay circuit which includes MOSFETs and flyback protection diodes as shown in the next section)

Relay Driver Circuit



The 2N7000 MOSFET was used as a switch to allow the arduino to control the relay's magnet as the arduino's output ports cannot handle the activation current required by the relay. And the Diode was used to protect the mosfet from flyback current due to the effect of inductance in the relay's magnet

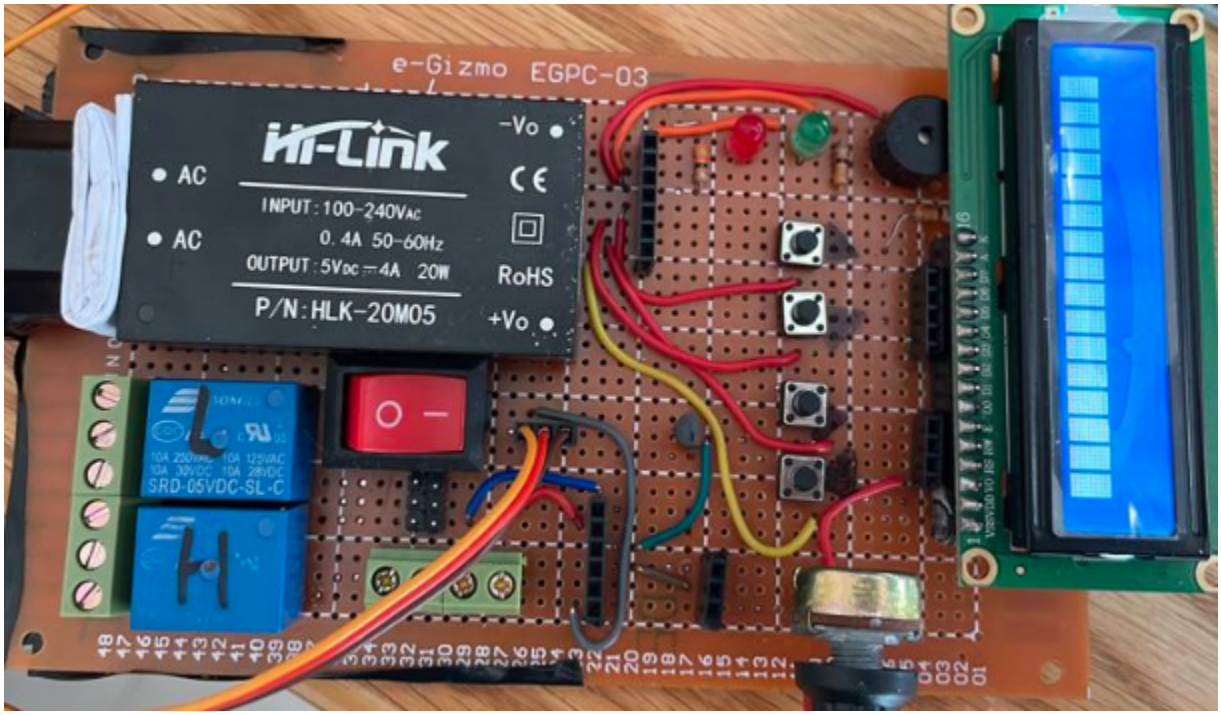
Electrical Design Considerations

Power Supply: a 20 Watt 5V power supply HLK-20M05 was used to maintain sufficient power to all the components, especially the 5W Heater and the Arduino.

Relay Outputs: Relay L was wired to output live 220V, Relay H was wired to output 5V (directly from power supply)

Backup Heater Timer: a 555 Timer was used to maintain a 50% heater duty cycle with a period of 6 seconds to avoid damage to the heater on the long run.

The Main Interfacing Board



IMPLEMENTATION DETAILS

The code is divided into 3 main sections:

1. The driver files
2. Global variables
3. The ISR(TIMER0_OVF_vect)
4. The main function

1. The driver files

Functions were abstracted and placed in separate files for ease of debugging and to achieve better overall coding experience.

- ➔ DIO Driver (dio.c)
- ➔ Analog to digital converter (ADC.c)
- ➔ LCD Driver (lcd.c)
- ➔ Servo Driver (servo.c)

  Note: The servo utilized phase correct PWM signal generation with a pulse width of 1-2 ms depending on direction and a total period of 50Hz which was controlled with the WGM mode 5 where the top was set to 156 (**OCRA**) and **OCR2B** was varied depending on required pulse width timing.

- ➔ Interrupt Timer (timer.c)

  Note: This was a normal mode timer (TIMER0) 8-bit overflow would trigger an interrupt flag which calls the ISR(TIMER0_OVF_vect) function (will be explained in the following section)

- ➔ Pushbuttons (pushbutton.c)
- ➔ Relay and Buzzer (relay.c)
- ➔ Load Cell (loadcell.c)

2. Global Variables

In the image below are all the variables used and their function. Global variables were used to ease communication between the ISR and the main function due to the fact that menu based navigation requires constant polling on the button press function and LCD display.

```
// GLOBAL VARIABLE DEFINITIONS

// MENU VARS
unsigned char key, c, tt;

// WEIGHT
unsigned short CURRENT_Weight = 60; // Current measured weight
unsigned char ALARM_Weight;         // This is set if weight exceeds threshold
unsigned short OCCUPANCY_Time = 0;  // Time current weight is above zero in seconds (for test purposes)
#define MAX_Weight 150              // if exceeded alarm weight is initiated

// TEMPERATURE
unsigned short BODY_Temp = 37;        // Body Temperature (sensor1 at ADC A2)
unsigned char ALARM_Fever = 0;        // This is set if body temp is above 37
unsigned short ROOM_Temp = 24;        // Room Temperature (sensor 2 at ADC A3)
unsigned short HEATER_Threshold = 10; // Temperature to be compared with ROOM_Temp for heater relay control, is set by LCD menu
unsigned char HEATER_Enable = 1;      // Is heater enabled? (done from lcd menu)
unsigned char HEATER_State = 0;       // If set, heater relay is turned on

// LAMP
unsigned char LAMP_Enable = 1; // Is lamp enabled (done from lcd menu)
unsigned char LAMP_State = 1;  // If set, lamp relay is turned on

// TODO: Decide on mode change logic

// IF MODE OLD NOT EQUAL NEW THEN MODE CHANGE, BASED ON MODE NEW
// DEFAULT MODE SITTING
unsigned char MODE_Old = 0; // 0 FOR SITTING 1 FOR SLEEPING
// 0 FOR SITTING 1 FOR SLEEPING
unsigned char MODE_New = 0; // 0 FOR SITTING 1 FOR SLEEPING

// TIMER VARS
unsigned char TIMER0_Counter = 0;  // counter to help increase the timer interrupt to 100ms
unsigned char TIMER0_Counter2 = 0; // second counter for 1 sec refresh rate
unsigned char TIMER0_Counter3 = 0;
#define TIMER0_Counter_100ms 6 // 16ms * 6 = 96ms
#define TIMER0_Counter_1s 64

```

3. The Timer0 Based ISR function

The main ISR function is divided into 3 Counter based if functions for 100ms 1s 10s delays

Each 100ms:

```
// INTERRUPT FUNCTION EACH 16ms
ISR(TIMER0_OVF_vect)
{
  TIMER0_Counter++;
  TIMER0_Counter2++;

  // ENTERS EACH 100ms
  if (TIMER0_Counter == TIMER0_Counter_100ms)
  {
    // START

    // ------------WEIGHT------------------//
    // Refresh current weight from adc
    CURRENT_Weight = LOADCELL_ReadWeight() / 3;
    // Check if max rated weight exceeded
    if (CURRENT_Weight > MAX_Weight)
    {
      ALARM_Weight = 1;
    }
    else if (CURRENT_Weight > 10) // if weight within operating range
    {
      OCCUPANCY_Time++; // each 100ms
      ALARM_Weight = 0;
    }
    else
    {
      OCCUPANCY_Time = 0; // if not used
      ALARM_Weight = 0;
    }

    //-------------TEMPERATURE-----------//
    BODY_Temp = (unsigned char)(((ADC_Read(2) * (5.0f / 1024) * 1000)) / 10); //
    ROOM_Temp = (unsigned char)(((ADC_Read(3) * (5.0f / 1024) * 1000)) / 10); //

    if (BODY_Temp > 37)
    {
      ALARM_Fever = 1;
    }
    else
    {
      ALARM_Fever = 0;
    }

    if ((ROOM_Temp < HEATER_Threshold) && HEATER_Enable)
    {
      HEATER_State = 1;
    }
    else
    {
      HEATER_State = 0;
    }

    // END of scope (100ms refresh)
    TIMER0_Counter = 0;
  }

```

Each 1 second (to prevent quick switching of relays)

```
// START OF 1SEC REFRESH

if (TIMER0_Counter2 == TIMER0_Counter_1s)
{

  // If a change in modes occurs the corresponding functions will be called
  if (MODE_Old != MODE_New)
  {
    if (MODE_New == 1)
    {
      SLEEP_Start();
    }
    if (MODE_New == 0)
    {
      WAKE_Start();
    }
  }
  // TEMPERATURE AND LIGHTING OUTPUTS
  if (HEATER_State == 1 && HEATER_Enable == 1)
  {
    RELAY_Heater(ON);
  }
  else
  {
    RELAY_Heater(OFF);
  }

  if (LAMP_State == 1 && LAMP_Enable == 1)
  {
    RELAY_Lamp(ON);
  }
  else
  {
    RELAY_Lamp(OFF);
  }
```

Each 10 Seconds (for alarms)

```
  // 10 second alarm trigger
  TIMER0_Counter3++;
  if (TIMER0_Counter3 == 10)
  {
    if (ALARM_Fever == 1 && ALARM_EN) // TODO: Add snooze counter
    {
      alarm_fever();
      ALARM_Fever = 0; // RESET ALARM FLAG
    }
    if (ALARM_Weight == 1 && ALARM_EN)
    {
      alarm_max_weight();
      ALARM_Weight = 0;
    }

    TIMER0_Counter3 = 0;
  }
```

4. The Main (Menu) Function

The main is comprised of two main sections:

1. Password Entry

   (in the main)

```
367     unsigned char mode = 5, Pass = 0, ff = 0;
368     lcd_setcursor(0, 4);
369     lcd_sendstring("WELCOME!");
370     _delay_ms(300);
371     LCD_SendCommand(1);
372     lcd_setcursor(0, 10);
373     lcd_sendstring("      For Login");
374     lcd_setcursor(1, 2);
375     lcd_sendstring("Press :  1");
376     key = choose(); // wait to check pressed button from the user
377     if (key == 1)
378     {
379       LCD_SendCommand(1);
380       while (Pass != 4) // will be in the loop while the password is not correct
381       {
382         ff = 0;
383         LCD_SendCommand(1);
384         lcd_sendstring("      USER : Hassan");
385         lcd_setcursor(1, 0);
386         lcd_sendstring("PASS : ");
387         while (ff != 4) // make password from 4 digit
388         {
389           key = choose();
390           LCD_SendData(key + '0'); // to recive correct number in ascii code
391           Pass += key;
392           ff++;
393           if (ff == 4)
394             _delay_ms(200);
395         }
396         ff = 0;
397         if (Pass != 4)
398         {
399           Pass = 0;
400           LCD_SendCommand(1);
401           lcd_sendstring("wrong pass");
402           lcd_setcursor(1, 3);
403           lcd_sendstring("try again");
404           _delay_ms(200);
405         }
406       }
```

2. Interactive Menu

Note: functions such as sleep1, sleep2.. Etc are defined separately (outside main) and serve as an abstraction to reduce the lines in the main function. And are where most of the global variables are updated/ read

```
// frame 1 in sleep mode LOADING
void sleep1(void)
{

  lcd_sendstring("      sleeping..");
  _delay_ms(2000);
  LCD_SendCommand(1);
  lcd_sendstring("      body temp:");
  lcd_send_number(BODY_Temp);
  lcd_setcursor(1, 0);
  lcd_sendstring("1:roomtmp ");
  lcd_sendstring("2:home ");
}
// frame 2 in sleep mode ROOM TEMPERATURE
void sleep2(void)
{
  // TODO: keep checking on ROOM_Temp variable
  LCD_SendCommand(1);
  lcd_sendstring("      room temp:");
  lcd_send_number(ROOM_Temp);
  lcd_setcursor(1, 0);
  lcd_sendstring("1:weight");
  lcd_sendstring(" 2:home ");
}
// frame 3 in sleep mode  CURENT WEIGHT
void sleep3(void)
{
  // TODO: keep checking on CURRENT_Weight variable
  LCD_SendCommand(1);
  lcd_sendstring("      weight:");
  lcd_send_number(CURRENT_Weight);
  lcd_setcursor(1, 0);
  lcd_sendstring("1:occ time");
  lcd_sendstring("2:home ");
}
```

```
void sleep4(void) // frame 4 in sleep mode SLEEP TIME (should be occupancy)
{
  // TODO: keep checking on OCCUPANCY_Time variable
  LCD_SendCommand(1);
  lcd_sendstring("      occupy time:");
  lcd_send_number(OCCUPANCY_Time);
  lcd_setcursor(1, 0);
  lcd_sendstring("2:home ");
}
void sit1(void) // frame 1 in sitting mode HOME MENU
{
  LCD_SendCommand(1);
  lcd_sendstring("      sitting..");

  _delay_ms(2000);
  LCD_SendCommand(1);
  lcd_sendstring("      options");
  lcd_setcursor(1, 0);
  lcd_sendstring("1:next");
  lcd_sendstring("   2:home ");
}
void sit2(void) // frame 2 in sitting mode HEATER ENABLE/DISABLE
{
  LCD_SendCommand(1);
  lcd_sendstring("      heating");
  lcd_setcursor(1, 0);
  lcd_sendstring("1:on");
  lcd_sendstring("  2:off ");
}
void sit3(void) // frame 3 in sitting mode HEATER ON SELECT TEMP
{
  c = 0;
  LCD_SendCommand(1);
  lcd_sendstring("      heat temp");
  lcd_setcursor(1, 0);
  lcd_sendstring("put temp:");
}
```

```
void sit3(void) // frame 3 in sitting mode HEATER ON SELECT TEMP
{
  c = 0;
  LCD_SendCommand(1);
  lcd_sendstring("      heat temp");
  lcd_setcursor(1, 0);
  lcd_sendstring("put temp:");
}
void sit4() // frame 4 in sitting mode LAMP ENABLE
{
  LCD_SendCommand(1);
  lcd_sendstring("      lamp enable");
  lcd_setcursor(1, 0);
  lcd_sendstring("1:on");
  lcd_sendstring("  2:off ");
}

unsigned char choose(void) // polling function to w8 user to press key
{
  do
  {

    key = PUSHBUTTONS_Read();

  } while (key == 0xff);
  return key;
}
```

And the menu control section....

```
while (mode == 5) // main function after user is allowed in
{
  LCD_SendCommand(1); // make user choose between 2 modes we have in our program
  lcd_sendstring("      1:for sleep mode");
  lcd_setcursor(1, 0);
  lcd_sendstring("2:for sit mode");
  mode = choose();
  LCD_SendCommand(1);
  if (mode == 1) // if user choose sleep mode
  {
    MODE_New = 1;
    sleep1();
    mode = choose();
    if (mode == 1) // user decides to proceed 1
    {
      sleep2();
      mode = choose();

      if (mode == 1) // user decides to proceed 2
      {
        sleep3();
        mode = choose();
        if (mode == 1) // user decides to proceed 3
        {
          sleep4();
          mode = choose();
          if (mode == 2) // user want to return home 3
          {
            mode = 5;
          }
        }
        else if (mode == 2) // user want to return home 2
        {
          mode = 5;
        }
      }
```

```
463             else if (mode == 2) // user want to return home 1
464             {
465 
466               mode = 5;
467             }
468           }
469 
470           else if (mode == 2) // if user choose home 0
471           {
472             mode = 5;
473           }
474         }
475         else if (mode == 2) // user choose sitting mode
476         {                   // last if condition
477           MODE_New = 0;     // enable sitting globally (will be read by interrupt)
478           sit1();
479           mode = choose();
480           if (mode == 1) // user want to proceed 1
481           {
482 
483             sit2();
484             mode = choose();
485             if (mode == 1) // user want to proceed 2
486             {
487               HEATER_Enable = 1;
488               LCD_SendCommand(1);
489               lcd_sendstring("heater on");
490               _delay_ms(200);
491               LCD_SendCommand(1);
492               sit3();
493               while (c != 2) // wait user to set temp then proceed auto to next step so here user has no option to return home
494               {
495                 key = choose();
496                 LCD_SendData(key + '0');
497                 HEATER_Threshold = 0;
498                 if (c == 0)
499                 {
500                   HEATER_Threshold = key * 10;
501                   tt = (key + '0') * 10;
502                 }
503                 _delay_ms(200);
504                 if (c == 1)
505                 {
506                   HEATER_Threshold += key;
507                   tt += (key + '0');
508                 }
509                 c++;
510                 if (c == 2)
511                 {
512                   _delay_ms(100);
513                 }
514               }
```

```
515             sit4(); // proceed to final frame in sitting mode
516             mode = choose();
517             if (mode == 1)
518             {
519               LAMP_Enable = 1;
520               LAMP_State = 1;
521               LCD_SendCommand(1);
522               mode = 5; // make mode 5 to return home after outing from this function
523               lcd_sendstring("lamp on");
524               _delay_ms(200);
525             }
526             else if (mode == 2)
527             {
528               LAMP_Enable = 0;
529               LAMP_State = 0;
530               LCD_SendCommand(1);
531               mode = 5; // make mode 5 to return home after outing from this function
532               lcd_sendstring("lamp off");
533
534               _delay_ms(200);
535             }
536         }
```

```
536             }
537             else if (mode == 2)
538             {
539               HEATER_Enable = 0;
540               LCD_SendCommand(1);
541               lcd_sendstring("heater off");
542               _delay_ms(200);
543               LCD_SendCommand(1);
544               sit4(); // proceed to final frame in sitting mode
545               mode = choose();
546               if (mode == 1)
547               {
548                 LAMP_Enable = 1;
549                 LAMP_State = 1;
550                 LCD_SendCommand(1);
551                 mode = 5; // make mode 5 to return home after outing from this function
552                 lcd_sendstring("lamp on");
553                 _delay_ms(200);
554               }
555               else if (mode == 2)
556               {
557                 LAMP_Enable = 0;
558                 LAMP_State = 0;
559                 LCD_SendCommand(1);
560                 mode = 5; // make mode 5 to return home after outing from this function
561                 lcd_sendstring("lamp off");
562                 _delay_ms(200);
563               }
564             }
565           }
566           else if (mode == 2)
567           {
568             mode = 5;
569           }
570         }
571       }
572     }
```