i. Front page

Coursework 2 – Report

Human Machine Interface

Coventry University

KH5048CEM

Embedded System Design and Development

Ziad Ayoub

CU1900705

BEng Electrical and Electronics Engineering

2021 - 2022

ii. Table of contents

i. Front page 1
ii. Table of contents 2
1.0 Introduction 3
2.0 System Design Architecture 4
2.1 Bill of Materials 5
3.0 Software Design, Testing, and Implementation 9
3.1 Hardware Code Software Explanation 9
3.1.1 Hardware Code Part 1 – Declarations 9
3.1.2 Hardware Code Part 2 – LowerLim Function 9
3.1.3 Hardware Code Part 3 – UpperLim Function 11
3.1.4 Hardware Code Part 3 – Main Function 12
3.2 Simulation 12
3.3 Software Implementation on GIT 13
3.4 Software Test Cases used to Validate 13
3.4.1 Temperature Change Cases 13
3.4.2 Temperature Lower Limit Verification Cases 13
3.4.3 Temperature Upper Limit Verification Cases 14

1.0 Introduction

The way a system communicates with an end user (Human) is a crucial part of any embedded system to make the system useable by people and makes the important information easily variable and understandable. In this project we used an LM35 sensor to read temperature. The temperature reading is taken and outputted on an LCD screen. An upper limit and lower limit for the temperature has been set and can be altered through a keypad by the user. If the temperature exceeds the limits a auditory buzzer sound is outputted.

Full demonstration of the system **on simulation (PICSIMLAB)** is shown in the video: https://elsewedyedu1-my.sharepoint.com/:v:/g/personal/ziad_ayoub_tkh_edu_eg/Ec1wTrcfiwdBjf2lY6t-SUIB8zyOZsVkb13iDJcPiANY1A?e=ofu7V6

Full demonstration of the system **on physical hardware** is shown in the video: https://elsewedyedu1-my.sharepoint.com/:v:/g/personal/ziad_ayoub_tkh_edu_eg/EfjHPXsj8jRFmREe2FkaPsgB4OHNX_i35ZdVcoUY9bngjQ?e=ENStqp

The software code (.c file) can be found through this link on GitHUB: https://github.com/Ziad-A/EmbeddedCW2

2.0 System Design Architecture

The system is based around the PIC18F47K42 Microcontroller chip paired with a EasyPIC V8 development board on the hardware version. The board is connected to the computer through a USB-C cable for power and debugging. Once the code was up and running, no interaction through the computer was necessary. When simulating the system on PICSIMLAB, the system was based on the PIC18F45K50.

An LCD is connected on the development board in an allocated slot and is used as a visual output. Through the pins on the development board the LM35 (temperature sensor), the buzzer, and the keypad were connected. The keypad was connected directly to Port D. Both the LM35 and buzzer were connected on a breadboard. The LM35 was connected to +5V, ground and pin RA1. The buzzer was connected to ground and pin RE0.

Pin RA1 was configured as analogue input and the signal was ran through an ADC (analog to digital converter). Port E, which encompasses pin RE0 was configured as digital output for the buzzer. Port D was configured as a digital input for the keypad. The rest of the pins and ports were configured as digital outputs for the LCD.

As external elements to test out the Temperature readings we used a heater and a fan as a cooler.

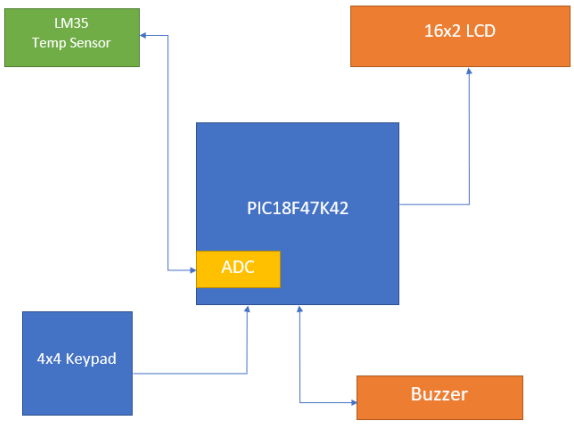

Figure - System Architecture Model

Full demonstration of the system **on simulation (PICSIMLAB)** is shown in the video: https://elsewedyedu1-my.sharepoint.com/:v:/g/personal/ziad_ayoub_tkh_edu_eg/Ec1wTrcfiwdBjf2lY6t-SUIB8zyOZsVkb13iDJcPiANY1A?e=ofu7V6

Full demonstration of the system **on physical hardware** is shown in the video: https://elsewedyedu1-my.sharepoint.com/:v:/g/personal/ziad_ayoub_tkh_edu_eg/EfjHPXsj8jRFmREe2FkaPsgB4OHNX_i35ZdVcoUY9bngjQ?e=ENStqp

2.1 Bill of Materials

Below is the list of materials used for the project, with the item name, description, quantity used, price per unit or per set (depending on how they're sold), the manufacturer (if applicable), and the link to the website where that component can be found and bought.

List of components: EasyPIC V8, mikroC PRO for PIC (software), PIC18F47K42 Microcontroller Chip, USB-C to USB-C cable, Jumper Wire, LM35 (Temp Sensor), LCD, Buzzer, Breadboard, Keypad. Heater, Fan.

Item: EasyPIC V8

Quantity: 1

Price per unit: $249

Description: It is a development board used to test and program compatible microcontrollers for embedded applications, based on 8-bit PIC microcontrollers (MCUs). The board is divide up into several sections including, LEDs, buttons, switches and mikroBUS socket.

Manufacturer: Mikro Elektronika

Link to website: https://www.mikroe.com/easypic

Item: mikroC PRO for PIC (software)

Quantity: 1

Price per unit: $269

Description: It is a software compatible with the EasyPIC V8 and a full-featured compiler for work and development on PIC microprocessors from Microchip Technology. It has an intuitive integrated development environment (IDE) and a complier. Also includes quick access to hardware and software libraries.

Manufacturer: Mikro Elektronika

Link to website: https://www.mikroe.com/mikroc-pic

Item: PIC18F47K42 Microcontroller Chip

Quantity: 1

Price per unit: $3.17

Description: DIP40 microchip (has 40 pins). Clock frequency: 64MHz. Program memory: 128kB. Supply voltage: 2.3 - 5.5V DC. Harvard 8bit Architecture.

Find the rest of the info attached in the data sheet:
https://www.tme.eu/Document/475e508ddf619bca87f4a27ea9509268/PIC18F47K42.pdf

Manufacturer: Microchip Technology

Link to website: https://www.tme.eu/en/details/pic18f47k42-i_p/8-bit-pic-family/microchip-technology/

Item: USB-C to USB-C 2.0 cable with adapter to USB 3.0 type A Male

Quantity: 1

Price per unit: $9.00

Description: USB 3.1 cable. USB-C male connectors on both sides. USB-C Female to USB-A Male adapter included. Compatible with USB 2.0 and USB 3.0 host slots as well. Cable length: 1.5m.

Manufacturer: Mikro Elektronika

Link to website: https://www.mikroe.com/usb-c-to-usb-c-20-cable-with-adapter-to-usb-30-type-a-male

Item: Arduino Jumper Wire Set

Price per set: 22 LE / 40 pieces

Description: Male to male jumper wires. Length: 20cm long and come in strips of 40. Current Rating: 1A.

Link to website: https://ram-e-shop.com/product/ph61-mm-20cm/

Item: Bread Board 630-Tie Point "BB-01"

Quantity: 1

Price per unit: 30 LE

Description: 300V/3-5A. 200 power tie points, 630 tie points. ABS Plastic material.

Link to website: https://ram-e-shop.com/product/bb01-bread-board/

Item: LM35dz "Temperature Sensor"

Quantity: 1

Price per unit: 25 LE

Description: Precision Centigrade Temperature Sensor. Reads a range between -55^{o}C and 150^{o}C. Operates From 4V to 30V. Data sheet: https://www.ti.com/lit/ds/symlink/lm35.pdf

Link to website: https://ram-e-shop.com/product/lm35dz/

Item: 1602A LCD 16x2 Char

Quantity used: 1

Price per set: 45 LE

Description: A display format 16 Character x 2 Line and display font of 5x8 dots. Power supply of 5V. Data sheet: https://www.openhacks.com/uploadsproductos/eone-1602a1.pdf

Link to website: https://store.fut-electronics.com/products/character-lcd-module-16-char-x2-lines?_pos=3&_sid=3b3feb325&_ss=r

Item: Buzzer 5V

Quantity used: 1

Price per set: 5 LE

Description: Rated frequency of 4,100Hz, sound pressure Level(30cm): 70dB, operating voltage of 5V-6V. Data sheet: http://www.farnell.com/datasheets/2171929.pdf

Link to website: https://ram-e-shop.com/product/buzzar6v/

Item: Keypad 4x4 Board

Quantity used: 1

Price per set: $9.00

Description: The Keypad 4x4 Board features 16 push buttons arranged in 4x4 matrix to form standard alphanumeric keypad.

Link to website: https://www.mikroe.com/keypad-4x4-board

3.0 Software Design, Testing, and Implementation

3.1 Hardware Code Software Explanation

The software is split up into 3 functions, a UpperLim function, a LowerLim function, and a Main function. Above the function declarations, are global variable declarations. The code can be found on GitHUB through this link: https://github.com/Ziad-A/EmbeddedCW2

3.1.1 Hardware Code Part 1 – Declarations

The first part of the code defines all the pin locations for the LCD which is necessary to initialize the LCD. After that it declares all variables needed throughout the code. All variables and initializations here are global variables.

```
1  //Ziad Ayoub CU1900705
2  //Assignment 2 Code - Hardwear Implimentation
3
4
5  //Declerations of LCD pins to initilize them
6  sbit LCD_RS at LATA3_bit;
7  sbit LCD_EN at LATB1_bit;
8  sbit LCD_D4 at LATB2_bit;
9  sbit LCD_D5 at LATB3_bit;
10 sbit LCD_D6 at LATB4_bit;
11 sbit LCD_D7 at LATB5_bit;
12
13 sbit LCD_RS_Direction at TRISA3_bit;
14 sbit LCD_EN_Direction at TRISB1_bit;
15 sbit LCD_D4_Direction at TRISB2_bit;
16 sbit LCD_D5_Direction at TRISB3_bit;
17 sbit LCD_D6_Direction at TRISB4_bit;
18 sbit LCD_D7_Direction at TRISB5_bit;
19
20
21 //Declerations of variables for temprature readings and temprature limits
22 int Temp;           //Creats the variable Temp as int
23 char TempStr[3];    //Creats varibale with length of 3 characters
24
25 int TempLimUpper = 60;      //Creats the variable for the temp upper limit as an integer and sets it to 60 degrees C
26 int TempLimLower = 20;      //Creats the variable for the temp lower limit as an integer and sets it to 20 degrees C
27
28 char TempLimUpperStr[3];    //Creats varibale with length of 3 characters
29 char TempLimLowerStr[3];    //Creats varibale with length of 3 characters
30
31
32 char keypadPort at PORTD;   //Specifes the location of the keypad at port D
33
34
35 Keypad_Init();      //Initializes the Keypad
36 int kp;   //Creats the varible kp (Key Pressed) as inteager
37
```

Figure - Main Code Declarations

3.1.2 Hardware Code Part 2 – LowerLim Function

The LowerLim function is a function that returns an integer which is the temperature lower limit value, and the function can be split up into 3 main parts. The first part is the increasing limit sub-function, the second part is the decreasing limit sub-function, and the last part is the LCD output and variable return.

If the 12th button on the keypad is pressed, it triggers the user to increase the value for the lower limit, by pressing a button that corresponds to a number on the keypad. If non-number button is pressed it is an invalid entry and waits for another click of a button. Whatever the button value

corresponds to, it is added to the limit. The range is from 0-9. 0 is an option if the user wants to cancel increasing the limit.

```
//This funcation returns the the value of the temprature lower limit as well as outputting it on the LCD.
//It also moniters specific button presses, which when pressed allows for the limit to be increased or decreased by a certain value.
//This value is determined by pressing a second button on the keypad.
int LowerLim(){

  //Pressed C button on keypad - Increase Lower Limit
  if (Keypad_Key_Click() == 12)
  {
    kp = 0;     //Sets key pressed to nothing
    //Keeps in the while loop until a keypad a number from 0-9 is pressed on the keypad.
    while(kp==0)
    {
      kp = Keypad_Key_Click();      //Sets the key pressed on the keypad
      //If the keys pressed are "A","B","C","D","#",or"*" the input is invalid so sets the key pressed back to nothing (kp=0)
      if((kp==4) || (kp==8) || (kp==12) || (kp==16) || (kp==13) || (kp==15))
      {kp=0;}
    }
    //Switch case for each of the possible numbers entered through the keypad.
    //Adds the respective number pressed to the Temp lower limit.
    switch (kp) {
       case 1: TempLimLower = TempLimLower+1; break;
       case 2: TempLimLower = TempLimLower+2; break;
       case 3: TempLimLower = TempLimLower+3; break;
       case 5: TempLimLower = TempLimLower+4; break;
       case 6: TempLimLower = TempLimLower+5; break;
       case 7: TempLimLower = TempLimLower+6; break;
       case 9: TempLimLower = TempLimLower+7; break;
       case 10: TempLimLower = TempLimLower+8; break;
       case 11: TempLimLower = TempLimLower+9; break;
       case 14: TempLimLower = TempLimLower+0; break;
    }
  }
```

Figure - LowerLim Function Part 1

If the 16th button on the keypad is pressed, it triggers the user to decrease the value for the lower limit, by pressing a button that corresponds to a number on the keypad. If non-number button is pressed it is an invalid entry and waits for another click of a button. Whatever the button value corresponds to, it is subtracted from the limit. The range is from 0-9. The number is only subtracted if the lower limit values is equal to or greater than the chosen value to avoid subzero limits. 0 is an option if the user wants to cancel decreasing the limit.

```
  //Pressed D button on keypad - Decrease Lower Limit
  else if (Keypad_Key_Click() == 16)
  {
    kp = 0;     //Sets key pressed to nothing
    //Keeps in the while loop until a keypad a number from 0-9 is pressed on the keypad.
    while(kp==0)
    {
      kp = Keypad_Key_Click();      //Sets the key pressed on the keypad
      //If the keys pressed are "A","B","C","D","#",or"*" the input is invalid so sets the key pressed back to nothing (kp=0)
      if((kp==4) || (kp==8) || (kp==12) || (kp==16) || (kp==13) || (kp==15))
      {kp=0;}
    }
    //Switch case for each of the possible numbers entered through the keypad.
    //Subtracts the respective number pressed to the Temp lower limit.
    //If the number chosen is larger than the value of the lower limit, it does not subtract it and keeps the limit as is.
    switch (kp) {
       case 1: if(TempLimLower>=1){TempLimLower = TempLimLower-1;} break;
       case 2: if(TempLimLower>=2){TempLimLower = TempLimLower-2;} break;
       case 3: if(TempLimLower>=3){TempLimLower = TempLimLower-3;} break;
       case 5: if(TempLimLower>=4){TempLimLower = TempLimLower-4;} break;
       case 6: if(TempLimLower>=5){TempLimLower = TempLimLower-5;} break;
       case 7: if(TempLimLower>=6){TempLimLower = TempLimLower-6;} break;
       case 9: if(TempLimLower>=7){TempLimLower = TempLimLower-7;} break;
       case 10: if(TempLimLower>=8){TempLimLower = TempLimLower-8;} break;
       case 11: if(TempLimLower>=9){TempLimLower = TempLimLower-9;} break;
       case 14: TempLimLower = TempLimLower-0; break;
    }
  }

  //Converts the temp lower limit into a string then writes it on to the LCD
  //"LLo" stands for "Limit "Limit Lower"
  ShortToStr(TempLimLower, TempLimLowerStr);
  Lcd_Out(2,4,TempLimLowerStr);
  Lcd_Out(2,1,"LLo:");

  //Returns temprature lower limit
  return TempLimLower;
}
```

Figure - LowerLim Function Part 2

After any possible increase or decreases to the value (or no alteration at all), the temperature lower limit value is taken, converted to a string and outputted onto the LCD. It then gets returned from the function.

```
102     //Converts the temp lower limit into a string then writes it on to the LCD
103     //"LLo" stands for "Limit "Limit Lower"
104     ShortToStr(TempLimLower, TempLimLowerStr);
105     Lcd_Out(2,4,TempLimLowerStr);
106     Lcd_Out(2,1,"LLo:");
107
108     //Returns temprature lower limit
109     return TempLimLower;
110 }
```

Figure - LowerLim Function Part 3

3.1.3 Hardware Code Part 3 – UpperLim Function

The UpperLim function is practically identical to the LowerLim function expect it is defined for the temperates' upper limit. It is split up into the same 3 segment configuration.

```
115 //This funcation returns the the value of the temprature upper limit as well as outputting it on the LCD.
116 //It also moniters specific button presses, which when pressed allows for the limit to be increased or decreased by a certain value.
117 //This value is determined by pressing a second button on the keypad.
118 int UpperLim(){
119
120   //Pressed A button on keypad - Increase Upper Limit
121   if (Keypad_Key_Click() == 4)
122   {
123     kp = 0;     //Sets key pressed to nothing
124     //Keeps in the while loop until a keypad a number from 0-9 is pressed on the keypad.
125     while(kp==0)
126     {
127       kp = Keypad_Key_Click();      //Sets the key pressed on the keypad
128       //If the keys pressed are "A","B","C","D","#",or"*" the input is invalid so sets the key pressed back to nothing (kp=0)
129       if((kp==4) || (kp==8) || (kp==12) || (kp==16) || (kp==13) || (kp==15))
130       {kp=0;}
131     }
132     //Switch case for each of the possible numbers entered through the keypad.
133     //Adds the respective number pressed to the Temp upper limit.
134     switch (kp) {
135        case 1: TempLimUpper = TempLimUpper+1; break;
136        case 2: TempLimUpper = TempLimUpper+2; break;
137        case 3: TempLimUpper = TempLimUpper+3; break;
138        case 5: TempLimUpper = TempLimUpper+4; break;
139        case 6: TempLimUpper = TempLimUpper+5; break;
140        case 7: TempLimUpper = TempLimUpper+6; break;
141        case 9: TempLimUpper = TempLimUpper+7; break;
142        case 10: TempLimUpper = TempLimUpper+8; break;
143        case 11: TempLimUpper = TempLimUpper+9; break;
144        case 14: TempLimUpper = TempLimUpper+0; break;
145     }
146   }
```

```
148   //Pressed B button on keypad - Increase Upper Limit
149   else if (Keypad_Key_Click() == 8)
150   {
151     kp = 0;     //Sets key pressed to nothing
152     //Keeps in the while loop until a keypad a number from 0-9 is pressed on the keypad.
153     while(kp==0)
154     {
155       kp = Keypad_Key_Click();      //Sets the key pressed on the keypad
156       //If the keys pressed are "A","B","C","D","#",or"*" the input is invalid so sets the key pressed back to nothing (kp=0)
157       if((kp==4) || (kp==8) || (kp==12) || (kp==16) || (kp==13) || (kp==15))
158       {kp=0;}
159     }
160     //Switch case for each of the possible numbers entered through the keypad.
161     //Subtracts the respective number pressed to the Temp upper limit.
162     //If the number chosen is larger than the value of the lower limit, it does not subtract it and keeps the limit as is.
163     switch (kp) {
164        case 1: if(TempLimUpper>=1){TempLimUpper = TempLimUpper-1;} break;
165        case 2: if(TempLimUpper>=2){TempLimUpper = TempLimUpper-2;} break;
166        case 3: if(TempLimUpper>=3){TempLimUpper = TempLimUpper-3;} break;
167        case 5: if(TempLimUpper>=4){TempLimUpper = TempLimUpper-4;} break;
168        case 6: if(TempLimUpper>=5){TempLimUpper = TempLimUpper-5;} break;
169        case 7: if(TempLimUpper>=6){TempLimUpper = TempLimUpper-6;} break;
170        case 9: if(TempLimUpper>=7){TempLimUpper = TempLimUpper-7;} break;
171        case 10: if(TempLimUpper>=8){TempLimUpper = TempLimUpper-8;} break;
172        case 11: if(TempLimUpper>=9){TempLimUpper = TempLimUpper-9;} break;
173        case 14: TempLimUpper = TempLimUpper-0; break;
174     }
175   }
176
177   //Converts the temp upper limit into a string then writes it on to the LCD
178   //"LHi" stands for "Limit "Limit Higher"
179   ShortToStr(TempLimUpper, TempLimUpperStr);
180   Lcd_Out(2,13,TempLimUpperStr);
181   Lcd_Out(2,10,"LHi:");
182
183   //Returns temprature upper limit
184   return TempLimUpper;
185 }
186
```

Figure - UpperLim Function

3.1.4 Hardware Code Part 3 – Main Function

The beginning of the main function defines all the ports sets the pins and ports either to digital or analog through ANSEL, 0 for digital and 1 for analogue. It also sets all ports as input or output through TRIS. 0 for output and 1 for input. The Analogue to Digital convertor (ADC) and the LCD get initialized. After that it enters the main while loop. Inside the while loop, the temperature is read and outputted onto the LCD. The upper and lower limits keep getting updated according to the inputs of the user and gathered from their respective functions. After that the temperature value gets compared to the values of the limits, if it exceeds the limits, a buzzer turns on.

```
void main() {
  ANSELA = 0x00000010;      //Sets port A as digital expect for pin 1 as analoug which is where the LM35 is connected
  TRISA = 0x00000010;       //Sets port A as output expect for pin 1 as input which is where the LM35 is connected

  ANSELB = 0;       //Sets port B as digital
  ANSELC = 0;       //Sets port C as digital
  ANSELD = 0;       //Sets port D as digital

  ANSELE = 0xff;    //Sets port E as analoug
  TRISE = 0;        //Sets for E as output

  ADC_Init();       //Initializes the analoug to digital converter

  Lcd_Init();           //Initializes the analoug to digital converter
  Lcd_Cmd(_LCD_CLEAR);              //Clear LCD display
  Lcd_Cmd(_LCD_CURSOR_OFF);         //Turns cursor off
  Lcd_Out(1, 1, "Temp Val: ");      //Prepare and output static text on LCD


  while(1){
    //Reads the value from the LM35 sensor and converts it into degrees celcuis.
    //Outputs the temp reading onto the LCD
    Temp = ADC_Read(1) * 0.489;
    WordToStr(Temp, TempStr);
    Lcd_Out(1,11,TempStr);


   //Gets the temp upper and lower limit for their respective funcation.
   //The funcations return the limit values as well as output the limit values on the LCD
    TempLimLower = LowerLim();
    TempLimUpper = UpperLim();


   //If the temp excued the limtits, it turns on the buzzer on which is conected to port E.
    if ((Temp >= TempLimUpper) || (Temp <= TempLimLower))
    {PORTE=0xff;}
    else
    {PORTE=0;}

  }
}
```

Figure - Main Function

3.2 Simulation

Due to errors occurring with the hardware during testing and development, alterative code was developed to test the overall system on a virtual alterative. The tool that was used was PICSIMLAB and the microcontroller that was used was the PIC18F45K50. The main differences is in the pin declarations as it is a different simulated development board and the increasing and decreasing of the limits is not as flexible and is only limited to incrementing and decrementing by 1. The full commented code can be found on GitHUB through the following link: https://github.com/Ziad-A/EmbeddedCW2

3.3 Software Implementation on GIT

To implement the code onto GitHUB, first a repository was created through my account. The repository name is “EmbeddedCW2” under the username “Ziad-A”. To push the code onto the repository, first GitBASH was downloaded and installed in order to use the Git Terminal. Through the terminal, the link between the repository “master” and computer “origin” were established.

The 2 software codes for the simulation and hardware (.c files) can be found through this link on GitHUB: https://github.com/Ziad-A/EmbeddedCW2

3.4 Software Test Cases used to Validate

3.4.1 Temperature Change Cases

The first thing that needs to be verified is to make sure that any temperature change registers correctly both on the temperature sensor (LM35) and on the LCD. To confirm this there are 2 test cases.

The first case is to bring a heater close by to the sensor, and to see if the temperature reading increases correctly and in a logical manner. The second case is to bring a fan close by to the sensor to cool it down, and once more, see if the temperature readings reacted correctly accordingly. In both the simulation and hardware testing implementation they reacted as expected.

3.4.2 Temperature Lower Limit Verification Cases

There are a total of 3 tests conducted to verify that the system’s temperature lower limit is functioning normally. The first one is to ensure that the limit increasing function runs as expected, and the second one to is to ensure that the limit decreasing function operates smoothly. The third test case to it confirm that a buzzer warning goes off when the temperature gets lower that the specified limit.

The first 2 cases were tested out by clicking on the specified buttons to increase and decrease the value. Additionally, for the hardware model using the keypad, each and every individual combination of keys was tested to make sure no errors are present.

The third case was tested by fixing a temperature reading and adjusting the temperature lower limit so that it is greater than the fixed temperature reading. The sub-system is confirmed to be working if the buzzer goes off.

3.4.3 Temperature Upper Limit Verification Cases

Similar to the temperature lower limit verification test cases, there are also a total of 3 tests conducted to verify that the system's temperature upper limit is functioning normally. The first one is to ensure that the limit increasing function runs as expected, and the second one to is to ensure that the limit decreasing function operates smoothly. The third test case to it confirm that a buzzer warning goes off when the temperature gets larger than the specified limit.

To test the first 2 cases, the same method was conducted as the first 2 cases of the lower limit verification where all button combinations were tried and verified.

For the third test case, the temperature reading was fixed, and the upper limit was manipulated and altered so that it is lower than the fixed temperature. We verify the system works once we hear the buzzer going off.