FIRE ALARM USING TEMPERATURE SENSOR

Team Members:

Student 1:
Full Name: Mohammed Rahman Sherif Khan Mohammad
Student id: 25042777

Student 2:
Full Name: Akshaya Srinivasaramesh Latha
Student id: 25042696

I. Introduction:

In this project, a fire alarm is built using Arduino Uno [1], which is interfaced with a temperature sensor and buzzer. The temperature sensor senses the heat generated due to burning or fire or the heatwaves. Buzzer connected to Arduino will give an alarm indication. Whenever fire triggered, it burns objects nearby and produces smoke. A fire alarm can also be triggered due to small smoke from candlelight, heatwaves generated from the radiator or oil lamps used in a household. Also, whenever heat intensity is high then also the alarm goes on. Buzzer or alarm is turned off whenever the temperature goes to normal room temperature and when the temperature level reduces. LCD display is also interfaced to the Arduino board. Arduino fire alarm system is an important system for industrial purposes as well as for household purposes. Whenever it detects fire or high temperature then it instantly alerts the user about the fire through the LCD display. For this purpose, Arduino Uno microcontroller is used which is from the Arduino family, LCD display and a temperature sensor . Also, the Arduino interfacing with an LCD display is done to display the status of the system whether the temperature is detected or not.

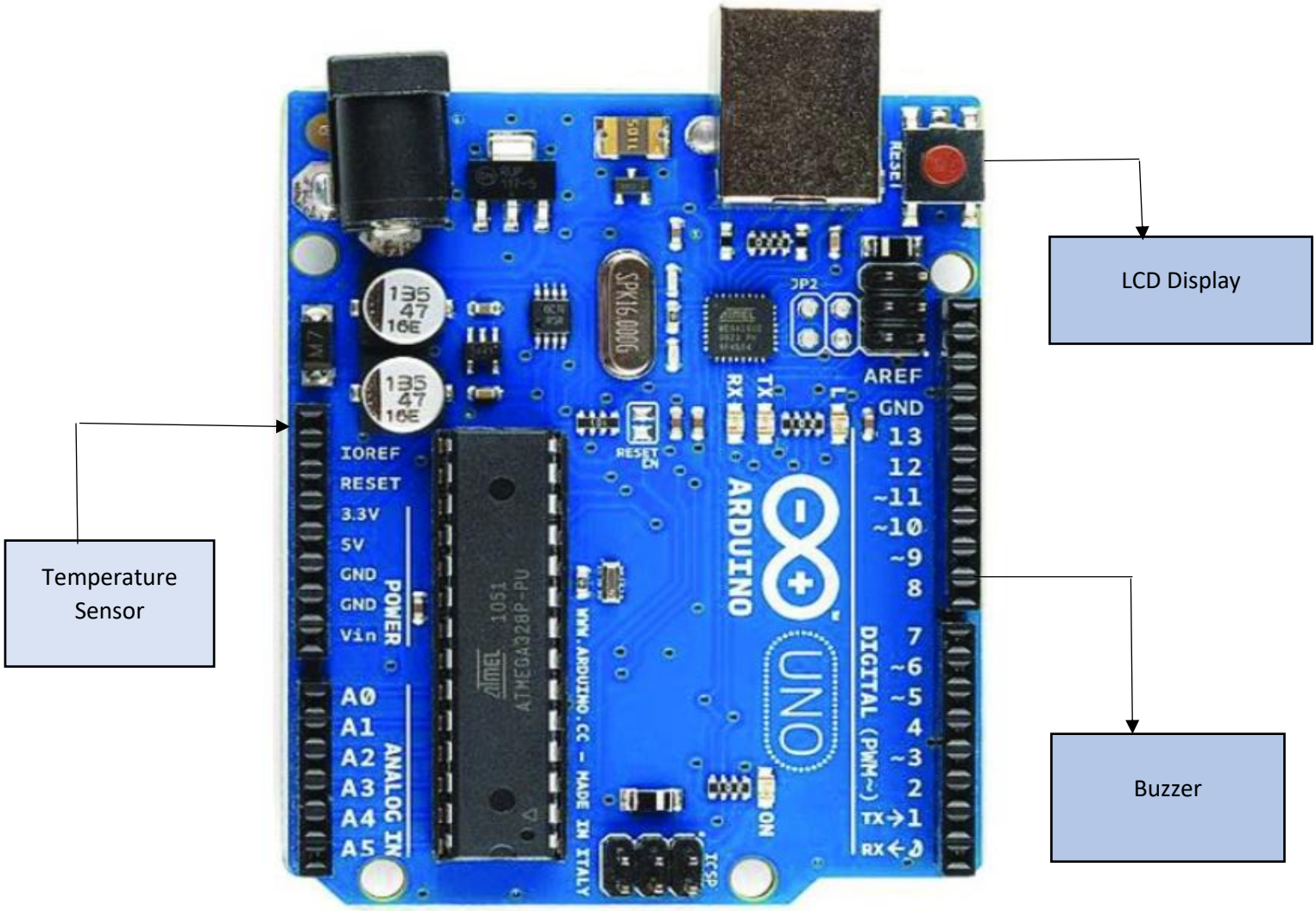


Figure 1: Signal sending and receiving block diagram using Arduino UNO

II. Aim:

The main aim of the project is to design and develop a fire alarm using temperature sensor.

III. Objectives:

i. Software Engineering model.
   - ✓ The preferred software engineering model is chosen based on the compatibility for the system and a brief explanation is given accordingly.

ii. System requirement specification.
   - ✓ The first objective is to identify the hardware and software requirements for the system to function.

iii. Programming the controller.
   - ✓ C programming language is used to program. Then the code should be compiled and uploaded using Arduino IDE software to the Arduino UNO board.

iv. Development of the system by connecting the circuit.
   - ✓ The controller, sensor, and other hardware, software components should be connected appropriately with proper power source.

v. Generating a test plan.
   - ✓ A test plan is generated for the system to function according to the plan.

vi. Testing the developed system.
   - ✓ The system will be simulated and the code will be uploaded using Arduino IDE software.

vii. Debugging the system.
   - ✓ The problems faced while testing the system is generated and then the errors are rectified by making the system function appropriately.

viii. Deployment of the system.
   - ✓ The system is debugged and then the rectified system is deployed and made to function.

ix. System maintenance.
   - ✓ The operation and the maintenance of the system.

i. Software Engineering model:

The Waterfall model is preferred to be employed. This sort of software engineering model is one that is preferred to implement since it is straightforward to comprehend and apply. The waterfall model is a static paradigm that takes a linear and sequential approach to system development, complete one task before moving on to the next. The waterfall approach divides projects into tasks such as requirement analysis, planning (estimation, scheduling, and tracking), modelling (analysis and design), construction (code and testing), and deployment (delivery, support and feedback)[2]. Furthermore, due to the rigidity of the model, it is simple to manage each phase, which has defined objectives and a review mechanism. In this approach, phases are processed and completed one at a time. Phases do not overlap. The waterfall paradigm works best for smaller projects with very well-defined and well understood requirements. Testing can begin as soon as executable software (even if it is only half developed) is available in waterfall models. After system requirements have been specified and implemented in testable programs, the majority of testing takes place. Commonly acknowledged issues including the inability to cope with change and the fact that flaws are frequently found too early in this software development process.

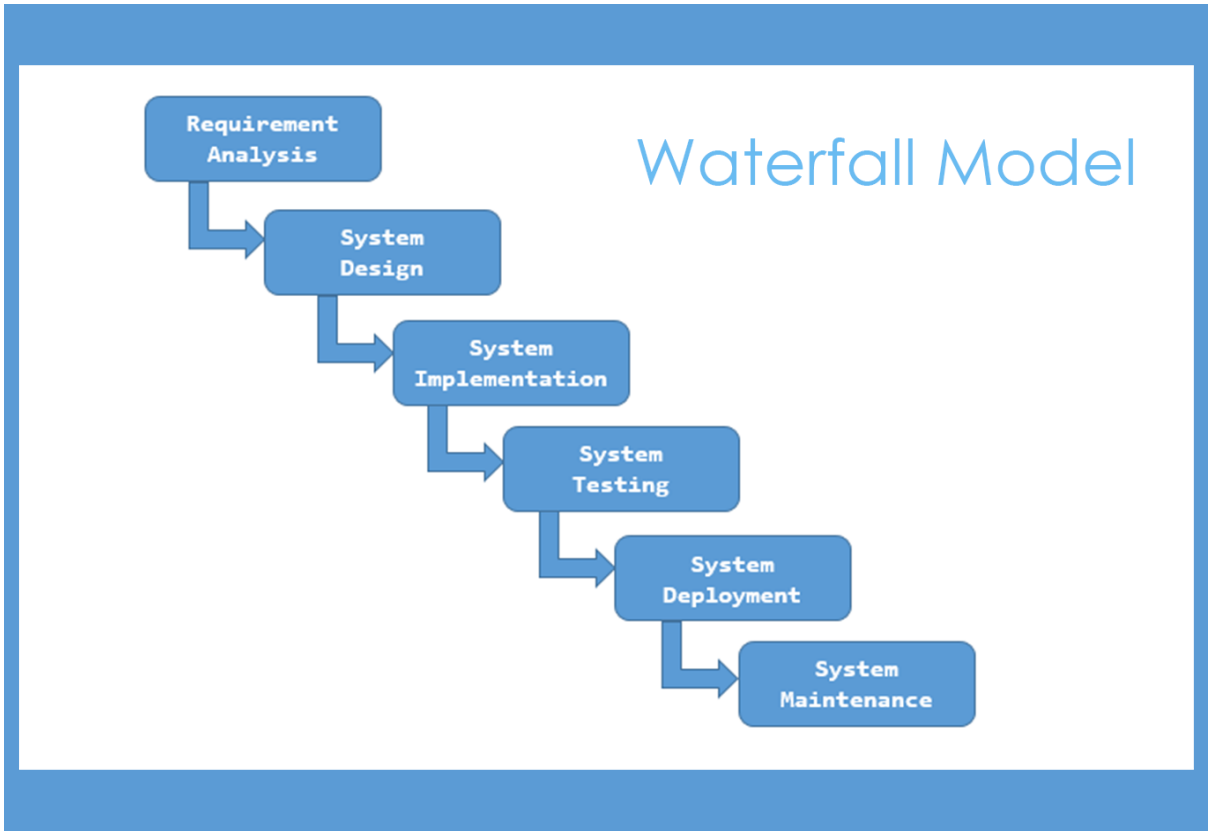


Figure2: General overview of the Waterfall model.

ii. System Requirement Specification:

❖ Hardware Requirements:

a) Arduino UNO:

Arduino Uno is a microcontroller board based on the ATmega328P (datasheet). It has 14 digital input/output pins.

b) Temperature sensor:

The Grove - Temperature Sensor uses a Thermistor to detect the ambient temperature. It's this characteristic that we use to calculate the ambient temperature. The detectable range of this sensor is -40 - 125ºC, and the accuracy is ±1.5ºC.

c) Jumper wires:

A jump is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype or test.

d) Grove Display kit:

The LCD (Liquid Crystal Display) is a type of display that uses the liquid crystals for its operation. Here, we will accept the serial input from the computer and upload the sketch to the Arduino. The characters will be displayed on the LCD.

e) Buzzer:

An arduino buzzer is also called a piezo buzzer. It is basically a tiny speaker that you can connect directly to an Arduino. You can make it sound a tone at a frequency you set. The buzzer produces sound based on reverse of the piezoelectric effect.

f) Bread board:

A breadboard, or protoboard, is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used when slicing bread. In the 1970s the solderless breadboard became available and nowadays the term "breadboard" is commonly used to refer to these

g) Arduino to USB cable:

A standard USB 1.1/2.0 Type B printer cable can be used for the Arduino Uno.

h) PC/Laptop:

We have used laptop to code the logics where we have installed Arduino software.

❖ Software Requirement:

a) Arduino IDE software:

The Arduino Integrated Development Environment (IDE) is a cross-platform application written in C and C++ functions for Windows, macOS, and Linux. It's used to write and upload programmes to Arduino-compatible boards, as well as other vendor development boards with the support of third-party cores.

iii. Programming of the controller:

❖ **Pseudocode:**

START

SET value to 0

WHILE

IF value > 300 DO

LCD print

BUZZER beep

END IF

LCD print value

ELSE IF value < 300 DO

LCD print

BUZZER stop beep

END IF

LCD print value

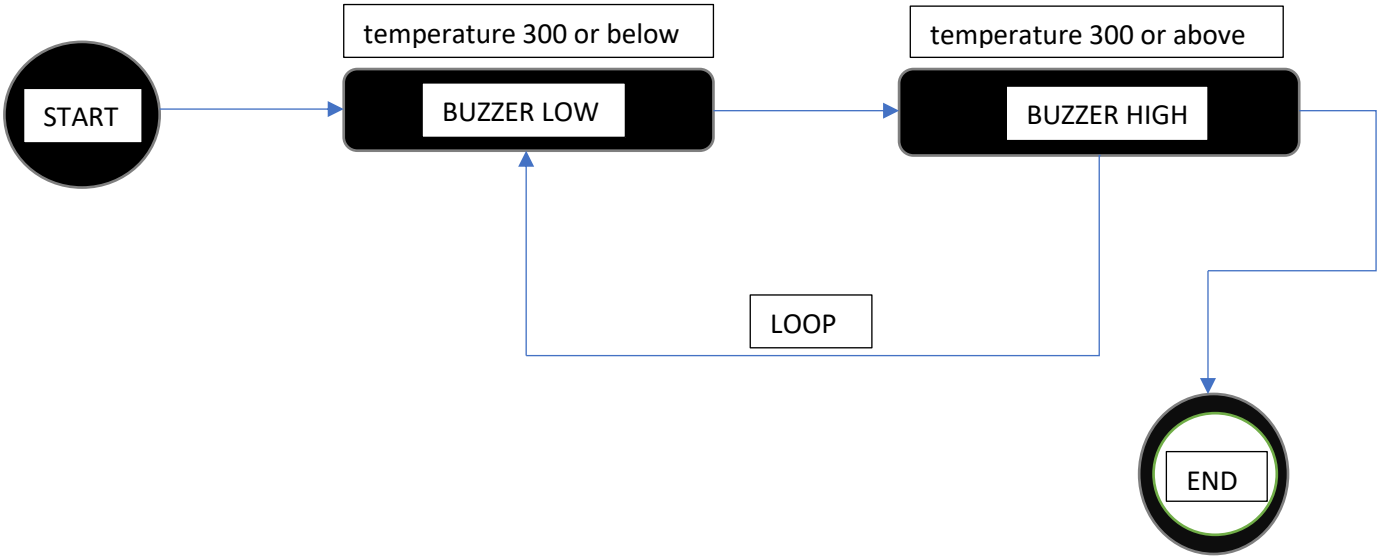
END WHILE

END

❖ **Pseudocode Explanation:**

The initial value of the temperature sensor is set to zero. If the value of the temperature sensor is higher than 300 then the buzzer beeps and the Liquid crystal display (LCD) displays the alert message and the value of the temperature sensor will be printed. If in case the value is below 300 then the buzzer doesn't beep and the LCD will print the message Feels good along with the value of the temperature sensor

❖ **State machine diagram:**



State Information Table:

STATE NAME	SPECIFICATIONS
LOW	The buzzer doesn't beeps. (*Figure 3*)
HIGH	The buzzer beeps. (*Figure 4*)

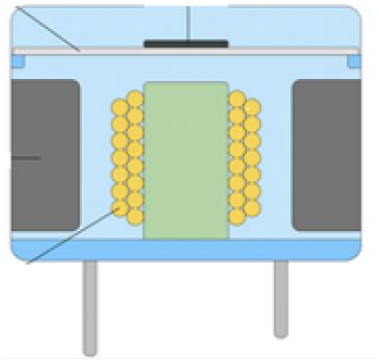


Figure 3: Buzzer LOW

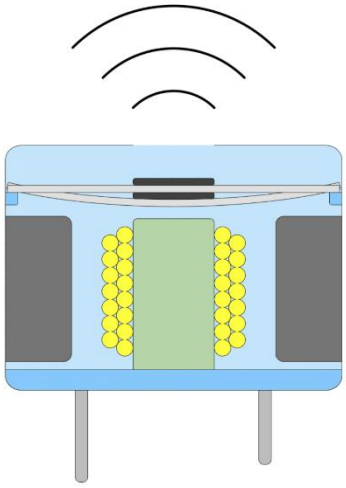


Figure 4: Buzzer HIGH

Transition Information Table:

TRANSITION NAME	SPECIFICATIONS
buzzer LOW	Temperature sensor value 300 or below.
buzzer HIGH	Temperature sensor value 300 or above.

❖ **Programming Code:**

```
#include <Wire.h> //importing libraries to communicate with the Arduino board
#include "rgb_lcd.h"
rgb_lcd lcd; // To change the LCD RGB backlight
const int colorR = 255; // Color code is inputted
const int colorG = 0;
const int colorB = 0;
int sensor = A0; // The temperature sensor is connected to the A0 analog input pin of the Arduino board
int buzzer = 2; // The signal pin of the buzzer is connected to the 2nd pin of the Arduino board
int value; // Input value
void setup() {
  // put your setup code here, to run once:
Serial.begin(9600); //The baud rate is 9600
pinMode(sensor, INPUT); // The input device is specified
pinMode(touch, INPUT); // The input device is specified
pinMode(2, OUTPUT); //The output pin value is specified
lcd.begin(16, 2); //Initializing the lcd screen
lcd.setRGB(colorR, colorG, colorB); // Color code is set
lcd.print("Current Temp...!"); //The lcd displays Current Temp
delay(500); // Delay seconds of the system is specified
}
void loop() {
  // put your main code here, to run repeatedly:
value = analogRead(7); //The analog input signal form pin 7 is read
if (value > 300){ // If the value read is greater than 300
lcd.setCursor(1, 1); //Position of the LCD cursor
lcd.print("Fire Alert:"); //The LCD displays Fire Alert along with the value above 300
digitalWrite(2, HIGH); // The value to the digital pin 2 is assigned as high
Serial.println(value);
```

```
}
lcd.setCursor(12, 1); //Position of the LCD cursor
lcd.print(value);
delay(500);
if (value < 300){ //If the value read is below 300
  lcd.setRGB(0, 0, 255); //Color code is set
  lcd.setCursor(1, 1); //The position of the LCD cursor
  lcd.print("Feels Good:"); //LCD displays feels good
  digitalWrite(2, LOW); //The value to the digital pin 2 is assigned as low
  Serial.println(value);
}
lcd.setCursor(12, 1);
lcd.print(value);
delay(500);
}
```

iv. Development of the system

The temperature around the sensor is detected. The temperature sensor has three pins which are the gnd, vcc, and signal. Both power pins are linked to 5v and gnd, and the signal pin is attached to the second pin on the Arduino board. The buzzer is made up of the very same three pins. As a result, it's coupled to the gnd and vcc in the same way. Similarly, the signal pin is linked to the Arduino board's second pin. The LCD display has also been appropriately linked to the vcc and gnd. In addition, the display's signal pins, such as sda and scl, have been linked to the Arduino board's A4 and A5. The circuit operates as follows, if the sensor data (temperature) exceeds the code's specified limit, the buzzer beeps, and the display module displays a fire alarm along with the current temperature value which is printed by the sensor. The temperature specified in the code is only detected in the event of a fire or an increase in temperature; if the temperature is below the limit, the buzzer will not beep, and the display will display information such as feeling good along with the current temperature

❖ **Output of the developed system:**

Figure 5: The LCD monitor display with a green background light when the Temperature is below 300.

Figure 6: The LCD monitor display with a red background light when the Temperature is above 300.

v. Test plan:

- ✓ Overall, the system function is based on hardware and the software requirements such as Arduino uno board, jumper wires, stepper motor, breadboard and Arduino cable. The connection is made on the Arduino board to the bread board by using jumper wires. The temperature sensor is connected to the Analog pin 0 and buzzer is connected to the 2nd pin of Arduino UNO. The Arduino UNO microcontroller is connected to the laptop through the Arduino cable. The programmed code is deployed in it.

- ✓ The pseudocode is written for the code to be programmed and then the pseudocode is explained. The state machine diagram is designed, based on the system's state and transition states which is written in the format of a table. From the written pseudocode and state machine diagram of the system, the programming code is written using the servo library which is in Arduino for the system to function appropriately and the code is uploaded in the Arduino board using Arduino cable and for simulating the system Arduino IDE software is used.

- ✓ And thus, the sensor will continuously read the data inputted while testing. If the sensor reads a value greater than 300 then the display module prints the message called "Fire alert" along with the temperature sensor value else if the temperature is moderate which is below 300 then the display will print a message "Feels good" along with the temperature sensor value. The system functions efficiently as per the commands employed in the programming code.

vi. Debugging the code:

The code which was written for the fire alarm to sense the temperature has been compiled successfully. Although the code worked but the system has not functioned as per the expectations. There is a logical error in the program but not compilation error. The below mentioned code which is present in the format of screenshot doesn't print the value of the sensor data(temperature). This is due to the curly braces present in the code. This has been fixed and the debugged code is found in testing the debugged code section below. The logical error in the program is pasted in the format of screenshot, this was taken while compiling the code.

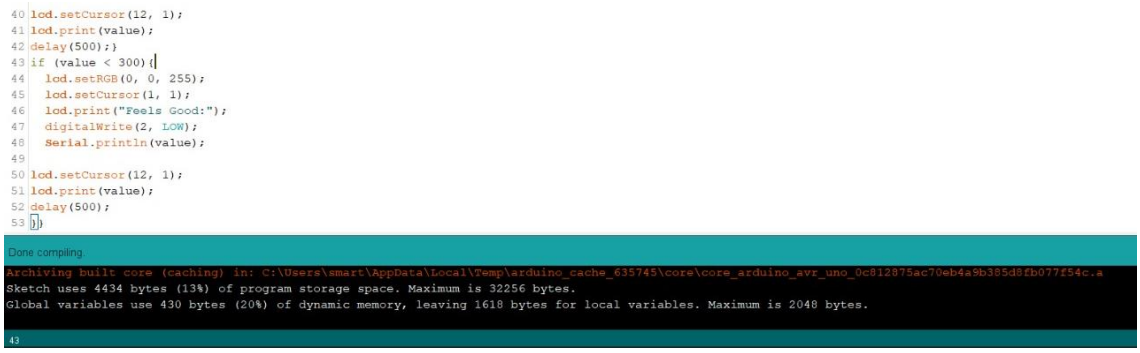


Figure 8: Screenshot of the logical error in the program source code.

vii. Testing the debugged code:

Then after debugging the code the code is been uploaded and tested again. Now, the code functions properly as per the given commands. The code had a logical error in it, which was spotted and rectified. Because of this reason the system didn't function properly. Thus, after this the system printed the values properly and also the buzzer responded and functioned quickly as soon as the temperature in the surrounding raised.

viii. Deployment of the system:

The functioning of the deployed system is added as a short video clip. Finally, while testing the system completely after debugging it. The system functions appropriately and responds quickly when the temperature soars up the fire alarm beeps and when the temperature drops down the Liquid crystal displays displays a message stating feels good in its monitor. The details of the functioning of the system is visualized in the format of a video clip which is attached below.

- **Video clip link:**

https://drive.google.com/file/d/1PX8bwS_8rHCROmEf1HpzbcZxf55rIkDc/view?usp=sharing

ix. System maintenance (Reflective discussion):

The coding part is completed and it is uploaded to the Arduino board. The circuit is designed appropriately but still the expected results hasn't been obtained from the system. So, first the connections in the circuit was cross verified which was given to the Arduino board. However, there was no mistake in the connections given on the circuit. And then the code was checked, that's the reason why the system went wrong. Even though there was a bug in the code but it was a logical error not a compilation error. Since two curly braces were added the system did not print the values which were read by the temperature sensor. Finally, after spotting the bug the error was rectified. From then onwards the system started to function according to the given commands through the code. At last after doing this step the expected output was obtained successfully which is already uploaded under the subheading output images in this report.

IV. Conclusion:

In a nutshell, the system is completely tested and it works appropriately. The temperature sensor detects the temperature when it goes above 300 and accurately alerts the buzzer and the buzzer starts to beep as soon as the signal is received from the temperature sensor. The LCD display also illustrates the alert immediately when the signal is sent by the temperature sensor. By implementing the software engineering model which is the waterfall model in the system has successfully completed all the phases in a step-by-step procedure. All the stages are completed by doing the following phases such as requirement analysis, designing the system, implementing, testing and then deploying of the system. Then a discussion is written about the maintenance of the system. The system works efficiently and the buzzer immediately starts to beep when there is a variation in the temperature value.

V. References:

[1] Mahzan, N.N., Enzai, N.I.M., Zin, N.M. and Noh, K.S.S.K.M. (2018). Design of an Arduino-based home fire alarm system with GSM module. *Journal of Physics: Conference Series*, 1019, p.012079.

[2] Tutorials Point (2019). *SDLC Waterfall Model*. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm.

- **Contribution Table:**

Mohammed Rahman Sherif Khan Mohammad	Akshaya Srinivasaramesh Latha
System Requirement Analysis	Introduction & conclusion
State Machine diagram	Software Engineering Model
Pseudocode	Aim & Objectives
Programming the controller	Pseudocode explanation & Commenting the program code
System maintenance	Test Plan
Testing the debugged system	Debugging the code
Deployment of the system	Deployment of the system
Development of the system	Development of the system