PPC Mini Project Report

Arduino-Based Health Monitoring Device with Temperature and Heartbeat Detection

1.Abstract:

This project presents the design and simulation of a health monitoring device using Arduino Uno. The system measures body temperature through an LM35 temperature sensor and simulates heart rate using a push button. The collected data is displayed on a 16×2 LCD and logged on the Serial Monitor in CSV format for analysis. An alert system comprising a buzzer and an LED is triggered whenever the temperature exceeds 38°C or the heart rate surpasses 100 BPM.

This prototype highlights how microcontroller-based devices can contribute to low-cost healthcare monitoring and can be extended into IoT-based systems for remote patient monitoring.

2. Introduction:

Monitoring vital health parameters such as temperature and heart rate is essential in early diagnosis and healthcare management. Traditional hospital devices are often costly and complex, but low-cost alternatives using microcontrollers like Arduino offer an accessible learning platform. This project implements a basic health monitor that measures body temperature and simulates heartbeat detection. Arduino processes the sensor signals, computes values in °C and BPM, and displays results on an LCD and Serial Monitor. In addition, the system includes an alert mechanism for abnormal readings.

3. Literature Review / Background Study:

- Wearable fitness trackers and hospital monitoring systems continuously log vitals, but they are expensive and proprietary.
- DIY Arduino-based monitoring projects often focus on either temperature or heart rate alone.
- Our project integrates **both vitals plus an alert system**, making it a compact and educational prototype.
- Unlike existing Tinkercad examples that only blink LEDs for sensor values, this system uses **LCD display + alert + logging**, providing a closer analogy to real devices.

4. Problem Statement:

There is a need for a simple, low-cost, and educational health monitoring prototype that can measure basic vital signs and provide alerts when values are abnormal. Current systems are either costly or too advanced for beginners. This project aims to bridge the gap by implementing a basic monitoring device using Arduino and C programming.

5. System Requirements:

Hardware

- Arduino Uno
- LM35 Temperature Sensor
- Push Button (for heartbeat simulation)
- 16×2 LCD + 10kΩ Potentiometer (contrast control)
- Piezo Buzzer
- Red LED + 220Ω Resistor
- Breadboard & Jumper Wires

Software

- Arduino IDE / Tinkercad Circuit Simulation
- C programming with Arduino libraries
- Excel / Google Sheets (for graphing logged data)

6. System Design:

Block diagram:

```
LM35 ---> A0 ----\
                 |--> Arduino Uno ---> LCD (16x2)
Button -> D3 ----/                  |
                          Alert System (Buzzer + LED)
                                 |
                        Serial Monitor (logging)
```

Flow chart:

Start

|

Read LM35 (Temp) & Button (Pulse)

|

Calculate BPM (time between beats)

|

Display Temp & BPM on LCD

|

Check thresholds:

- Temp > 38°C ?

- BPM > 100 ?

|

If abnormal → Activate Buzzer + LED + Show "ALRT"

Else → Normal Display

|

Log data (Time, Temp, BPM) to Serial Monitor

|

Repeat

7. Implementation:

- **LM35** connected to A0 for analog temperature readings.
- **Button** connected to pin 3, configured with INPUT_PULLUP, simulates heartbeat input.
  LCD wired in 4-bit mode (RS=12, E=11, D4=10, D5=9, D6=8, D7=7).
  Buzzer connected to pin 4, **LED** to pin 5 via 220Ω resistor.
- Arduino calculates temperature and BPM, displays on LCD, and logs to Serial Monitor.
- Alert activates when thresholds are exceeded.

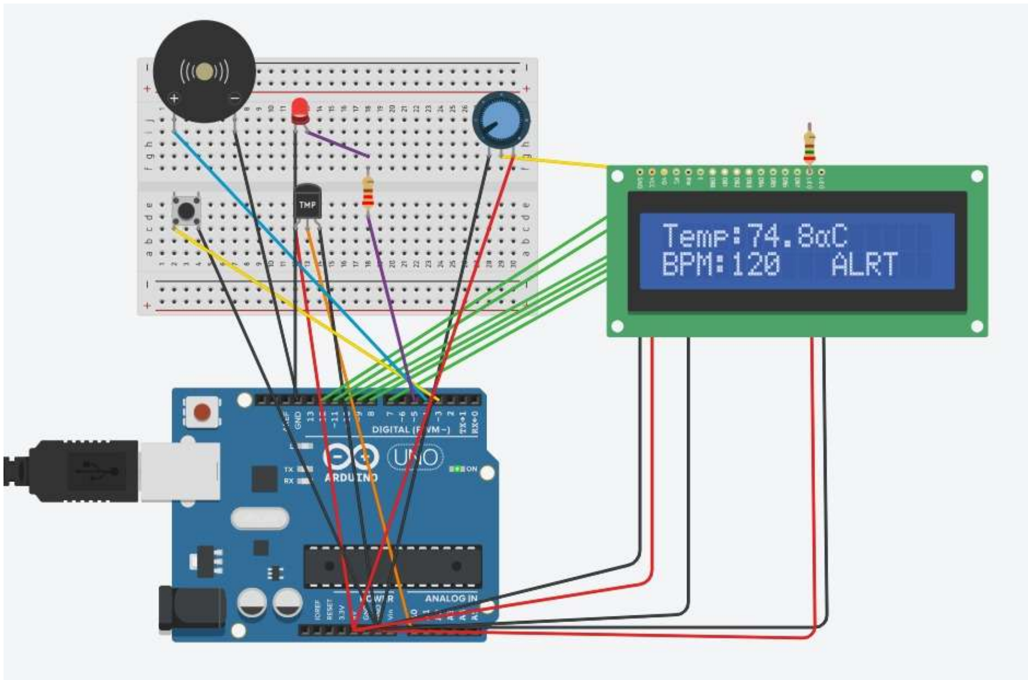
Complete code:

```
#include <LiquidCrystal.h>

// Pins
const int TEMP_PIN = A0;     // LM35 temperature sensor
const int PULSE_PIN = 3;     // Push button for pulse
const int BUZZER_PIN = 4;    // Buzzer
const int LED_PIN = 5;       // Red LED
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);

// BPM smoothing
const int BPM_HISTORY = 4;
float bpmHistory[BPM_HISTORY];
int bpmIndex = 0;
int bpmCount = 0;

// Timing & state
unsigned long lastBeatTime = 0;
int lastButtonState = HIGH;   // Button idle = HIGH
const unsigned long MIN_BEAT_INTERVAL = 250UL; // ignore >240 BPM

// Temperature timing
unsigned long lastTempMillis = 0;
const unsigned long TEMP_READ_INTERVAL = 1500UL;

void setup() {
  pinMode(PULSE_PIN, INPUT_PULLUP);  // Button wired to GND
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);

  lcd.begin(16, 2);
  Serial.begin(9600);

  lcd.print("Health Monitor");
  delay(1000);
  lcd.clear();

  // Init BPM history
  for (int i = 0; i < BPM_HISTORY; i++) bpmHistory[i] = 0;
}
```

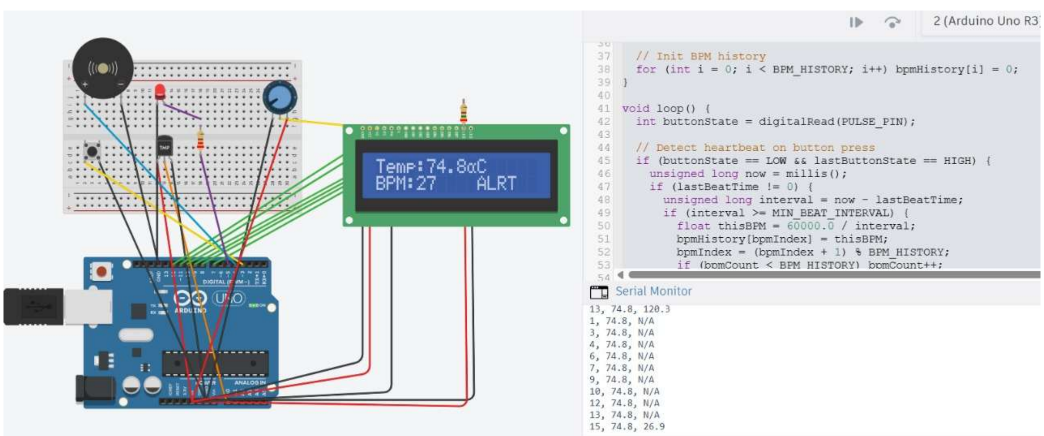
```
void loop() {
  int buttonState = digitalRead(PULSE_PIN);

  // Detect heartbeat on button press
  if (buttonState == LOW && lastButtonState == HIGH) {
    unsigned long now = millis();
    if (lastBeatTime != 0) {
      unsigned long interval = now - lastBeatTime;
      if (interval >= MIN_BEAT_INTERVAL) {
        float thisBPM = 60000.0 / interval;
        bpmHistory[bpmIndex] = thisBPM;
        bpmIndex = (bpmIndex + 1) % BPM_HISTORY;
        if (bpmCount < BPM_HISTORY) bpmCount++;
      }
    }
    lastBeatTime = now;
  }
  lastButtonState = buttonState;

  unsigned long nowMillis = millis();
  if (nowMillis - lastTempMillis >= TEMP_READ_INTERVAL) {
    lastTempMillis = nowMillis;

    // Average BPM
    float avgBPM = 0;
    if (bpmCount > 0) {
      for (int i = 0; i < bpmCount; i++) avgBPM += bpmHistory[i];
      avgBPM /= bpmCount;
    }

    // Read LM35 temperature
    int raw = analogRead(TEMP_PIN);
    float voltage = raw * (5.0 / 1023.0);
    float tempC = voltage * 100.0;

    // ---- ALERT SYSTEM ----
    bool alert = false;
    if (tempC > 38.0 || (bpmCount > 0 && avgBPM > 100)) {
      alert = true;
    }
    if (alert) {
```

```
    digitalWrite(BUZZER_PIN, HIGH);
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(BUZZER_PIN, LOW);
    digitalWrite(LED_PIN, LOW);
  }

  // ---- SERIAL LOGGING ----
  // Format: time (s), temperature, BPM
  Serial.print(millis() / 1000);
  Serial.print(", ");
  Serial.print(tempC, 1);
  Serial.print(", ");
  if (bpmCount > 0) Serial.println(avgBPM, 1);
  else Serial.println("N/A");

  // ---- LCD Display ----
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Temp:");
  lcd.print(tempC, 1);
  lcd.print((char)223); // degree symbol
  lcd.print("C");

  lcd.setCursor(0, 1);
  lcd.print("BPM:");
  if (bpmCount > 0) lcd.print(avgBPM, 0);
  else lcd.print("--");

  if (alert) {
    lcd.setCursor(10, 1);
    lcd.print("ALRT");
  }
 }
}
```

8. Results and Output:

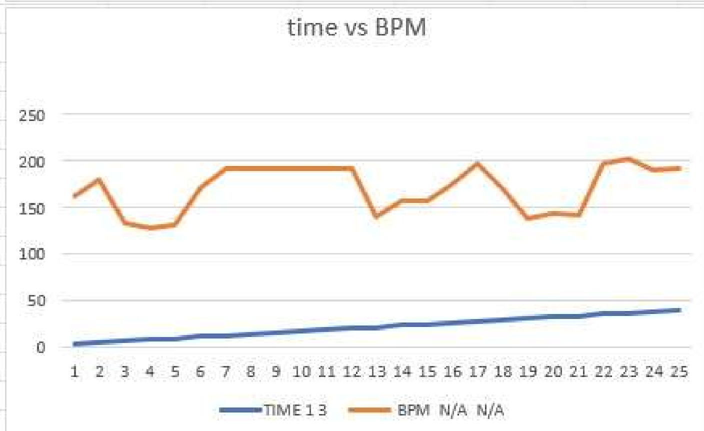


Serial monitor output:



Graphs generated from the real Serial Monitor data:

TIME	Temp(Â°(	BPM
1	74.8	N/A
3	74.8	N/A
4	74.8	163
6	36.2	179.9
7	34.2	133.6
9	19.1	127.7
10	19.1	131.9
12	19.1	172.1
13	19.1	192.4
15	19.1	192.4
16	19.1	192.4
18	9.8	191.6
19	47.9	191.6
21	34.2	191.6
22	34.2	141.2
24	34.2	157.8
25	34.2	157.8
27	34.2	174.6
28	34.2	197.5
30	34.2	170.3
31	34.2	138.1
33	34.2	144.5
34	34.2	142.7
36	34.2	197.2
37	34.2	203.2
39	34.2	190.7
40	34.2	192.8



9. Discussion and Analysis:

The project successfully simulated a health monitoring device in Tinkercad. Temperature and BPM were measured, displayed, and logged. The alert system correctly activated when thresholds were exceeded.

Challenges faced: Tinkercad does not support real pulse sensors, so a push button was used for heartbeat simulation. Adjusting LCD contrast and correct pin wiring were critical for output display.

10. Applications and Future Scope:

Applications:

- Educational tool for learning embedded systems.
- Prototype for low-cost health monitoring in rural or resource-limited areas.
- Demonstration of IoT healthcare systems.

Future Scope:

- Replace button with a real pulse sensor.
- Add Bluetooth/Wi-Fi for remote monitoring.
- Store data on SD card or cloud.
- Add more sensors (blood oxygen, humidity, motion).
- Implement different alert tones based on condition severity.

11. Conclusion:

This mini project successfully demonstrated a health monitoring device using Arduino. The system measured temperature and simulated BPM, displayed results on an LCD, logged them for analysis, and triggered alerts on abnormal conditions.
Key learnings include sensor interfacing, analog-to-digital conversion, interrupt-based timing, LCD handling, and data logging. This project highlights how Arduino can be applied to healthcare and extended into real IoT-based monitoring systems.

12. References

- Arduino Documentation: https://www.arduino.cc
- Tinkercad Circuits: https://www.tinkercad.com
- Datasheet: LM35 Precision Temperature Sensor
- TutorialsPoint / Arduino Reference

13.Tinkercad link:

link to tinker the project