Student Name

Mayar Waleed Nawas

Student ID

120220147

Title

Heart Rate Monitoring System Using Arduino

Supervisor To

Dr. Ahmed AbuMsameh

1. Project Overview

This project demonstrates a heart rate monitoring system simulated and implemented using **Tinkercad**, a cloud-based platform for Arduino circuit design and programming. The system tracks heart rate data (**BPM**), displays the readings on an **LCD screen**, and provides audio-visual alerts if the heart rate is outside the safe range. Additionally, an oscilloscope is integrated to **visualize** heart rate conditions through waveforms.

---

2. Objectives

- To monitor and display heart rate in real-time.
- To alert the user if the heart rate deviates from the defined safe range.
- To provide a foundation for integrating more advanced healthcare monitoring features.
- Use Tinkercad for visualizing hardware interactions and coding.
- Provide alerts for abnormal heart rate values to ensure user safety.
- Visualize heart rate patterns using an **oscilloscope**.

---

3. Components

3.1 Hardware:

- **Arduino Uno**: The microcontroller board used to execute the code and manage sensors and outputs.
- **LCD Display (16x2)**: Displays heart rate and system messages.
- **Piezo Buzzer**: Provides an audio alert if heart rate is abnormal.
- **LED/Indicator**: Gives a visual alert alongside the buzzer.
- **Oscilloscope**: Visualizes the heart rate as waveform.
- **Resistors**: Required for connections and component stability.

3.2 Software:

- Tinkercad: Used to simulate the circuit and upload code virtually.
- Adafruit_LiquidCrystal Library: Manages the LCD display.
- Arduino C-like Code: The programming language used in Tinkercad for coding and simulation.

4. Functional Requirements

Heart Rate Simulation:

Generate a heart rate value between predefined ranges (e.g., 60–180 BPM).

Alerts:

Display messages based on the heart rate range:

Safe Range: 120–160 BPM.

Low or High Heart Rate: Trigger alerts.

Oscilloscope Visualization:

Display a zigzag waveform for safe heart rates.

Display a flat line for critical heart rates.

5. System Workflow

Initialization:

Configure pins and initialize peripherals such as the LCD screen, buzzer, and oscilloscope.

Heart Rate Simulation:

Generate random heart rate values within a specified range to mimic real-time data.

Data Processing:

Check if the simulated heart rate is within the safe range.

Output:

Display the heart rate and status on the LCD.

Activate alerts (buzzer, LED, or oscilloscope) for abnormal heart rates.

Waveform Display:

Show zigzag waves for safe heart rates.

Show flat lines for critical heart rates.

Simulation Repeat:

Update values and system status every second.

6. Code Explanation

1. Setup:

Configures pins, initializes the LCD, and prepares the simulation:

```
void setup() {
    pinMode(TONE_PIN, OUTPUT); // Buzzer
    pinMode(ALERT_PIN, OUTPUT); // Alert LED
    pinMode(SIGNAL_PIN, OUTPUT); // Oscilloscope Signal
    lcd.begin(16, 2); // Initialize LCD
    lcd.print("BPM is: ");
}
```

2. Simulated Heart Rate:

Generates a random heart rate for testing purposes:

```
simulatedHeartRate = random(60, 180);
bpm = simulatedHeartRate;
```

3. Alert Logic:

Processes the heart rate and triggers alerts if needed:

```
if (bpm >= HEART_ALERT_LOW && bpm <= HEART_ALERT_HIGH) {
    lcd.print("Safe");
    digitalWrite(ALERT_PIN, LOW); // Turn off LED
    noTone(TONE_PIN); // Disable buzzer
    showSafeWave(); // Display zigzag wave
} else {
    digitalWrite(ALERT_PIN, HIGH); // Turn on LED
    tone(TONE_PIN, 2000, 500); // Activate buzzer
    lcd.print(bpm < HEART_ALERT_LOW ? "Heart Rate Low" : "Heart Rate High");
    showCriticalWave(); // Display flat line
}
```

4. Oscilloscope Waveforms:

Safe Heart Rate (Zigzag):

```
void showSafeWave() {
    for (int i = 0; i < 5; i++) {
        digitalWrite(SIGNAL_PIN, HIGH);
        delay(100);
        digitalWrite(SIGNAL_PIN, LOW);
        delay(100);
    }
}
```

Critical Heart Rate (Flat Line):

```
void showCriticalWave() {
    digitalWrite(SIGNAL_PIN, HIGH);
    delay(1000);
    digitalWrite(SIGNAL_PIN, LOW);
    delay(1000);
}
```

5. Loop:

Continuously updates and processes heart rate values:

```
void loop() {
    bpm = random(60, 180);
    lcd.clear();
    lcd.print("BPM is: ");
    lcd.println(bpm);
    checkHeartRate();
    delay(1000); // Wait for 1 second
}
```

7. Integration Notes for Oscilloscope

Connecting the Oscilloscope:

Connect the SIGNAL_PIN (Pin 9) to the oscilloscope's Channel A input.

Connect the oscilloscope's GND terminal to the Arduino's GND.

Calibration:

Adjust the oscilloscope's time/div and volts/div settings to clearly view zigzag and flat-line patterns.

---

8. Conclusion

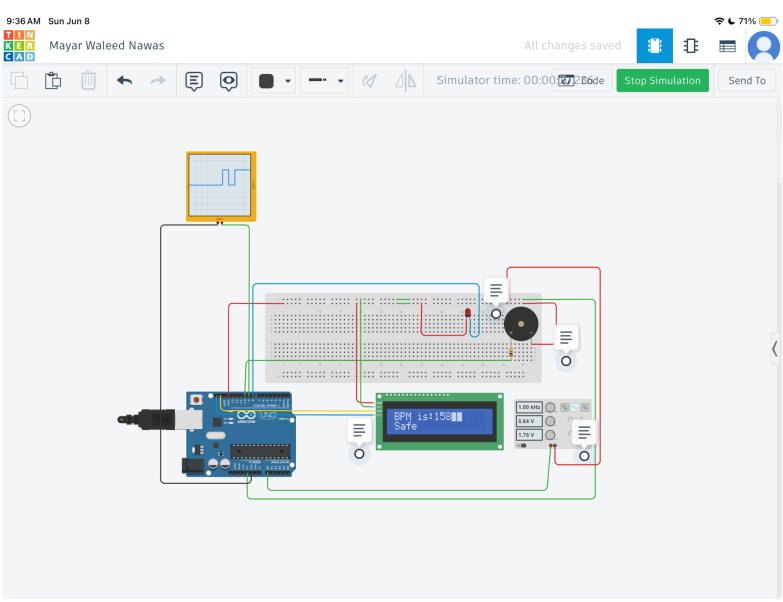
This project provides a foundational system for real-time heart rate monitoring using Arduino. By integrating additional sensors and communication modules, it can be developed into a comprehensive healthcare monitoring solution.

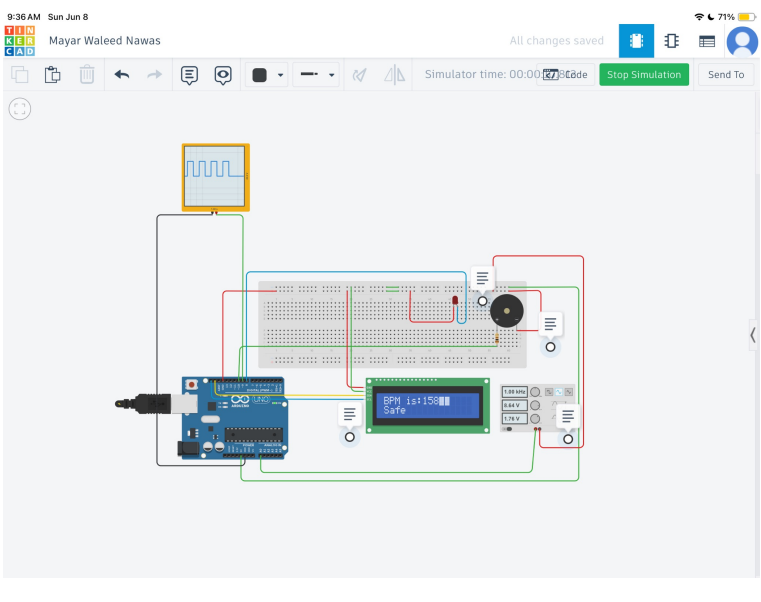
---

9. Appendix

Link Tinkercad : https://www.tinkercad.com/things/lr4w46S3vpJ-mayar-waleed-nawas

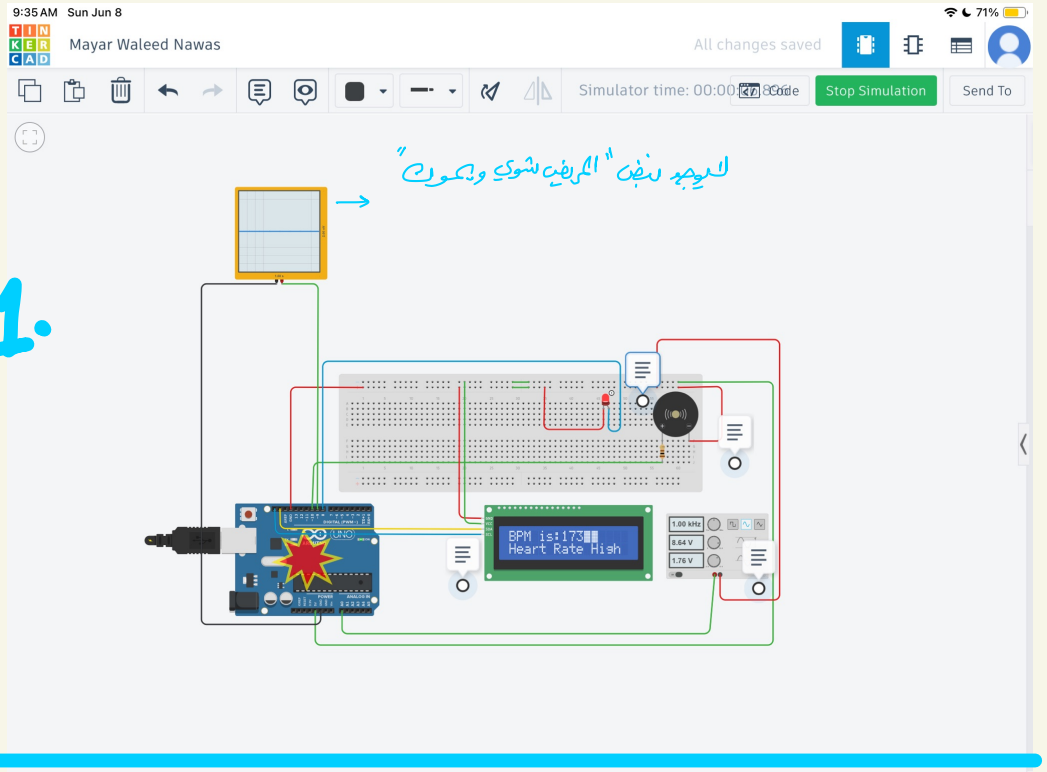
Link Video At Google Drive : https://drive.google.com/file/d/19YApv7CdLMPp5m0vQt-kEDIO2qtOJJ3P/view?usp=drivesdk



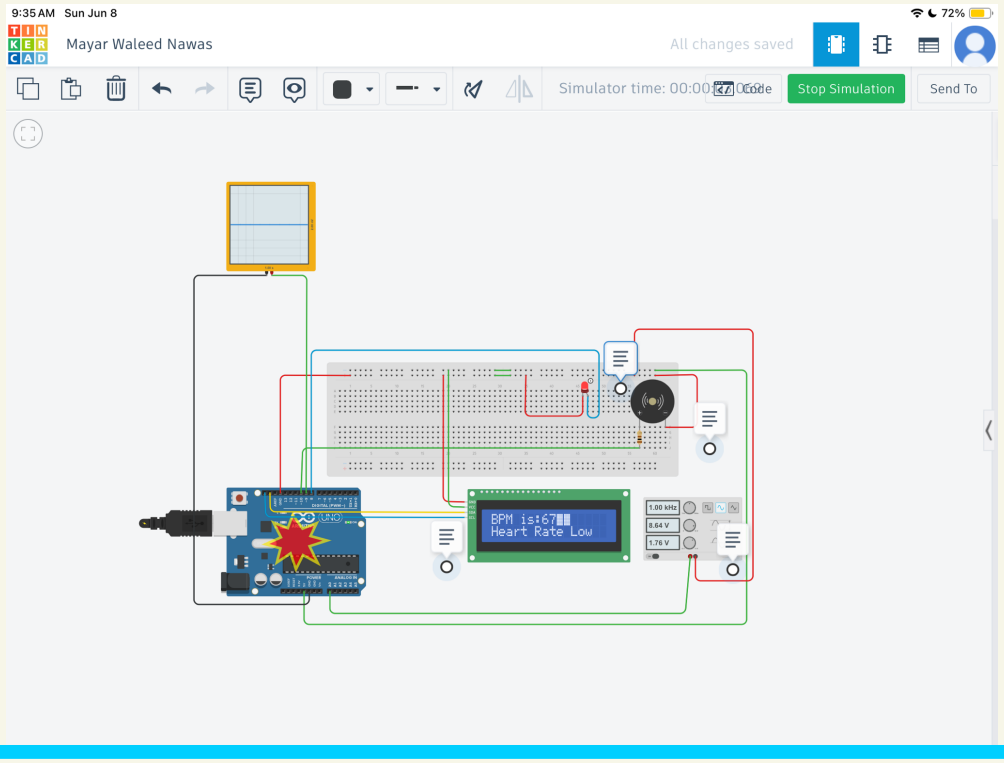


Danger Case

1.



2.



3.