Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Embedded Systems Lab Report

on

"Heart Rate Monitor"

[Code No: COMP 306]

Submitted by

Abiral Adhikari (02)

Prashant Manandhar (30)

Samir Wagle (60)

Reewaj Khanal (61)

Submitted to

Prof. Sudan Jha

Department of Computer Science and Engineering

Submission Date: 16/06/2024

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

Heart rate monitoring is an essential function in both personal health management and clinical diagnostics, providing crucial information about an individual's cardiovascular health. This project focuses on developing an embedded system that simulates a heart rate monitor using Proteus for simulation and an Arduino microcontroller for implementation. A pulse sensor is utilized to detect the heart rate, and the data is processed and displayed on an LCD screen. The system also incorporates potentiometer to control the heart rate signal during the simulation. This simulation aims to validate the design and functionality of the heart rate monitoring system before practical deployment, ensuring accuracy and reliability in real-world applications. By leveraging the capabilities of Proteus and Arduino, this project demonstrates the integration of hardware and software in creating an effective heart rate monitoring solution

1.2 Objectives

The basic purpose of this project is to develop a reliable and accurate heart rate monitoring system using Proteus for simulation and an Arduino microcontroller for real-time monitoring. The objectives of the project are:

- To design and simulate a heart rate monitoring system using Proteus software.
- To implement the heart rate monitoring system using an Arduino microcontroller and a pulse sensor.
- To process and display the heart rate data on an LCD screen for easy readability.
- To ensure the accuracy and reliability of the heart rate measurements through Simulation in Proteus.

Chapter 2 Component Breakdown

1. **Arduino:** The Arduino is a microcontroller board that serves the mind of the challenge, processing sensor facts and controlling the LCD display screen. It provides the programming interface to bring all the components together.
2. **Heart Rate Sensor:** The heart rate sensor measures the user's pulse by detecting changes in blood volume using infrared light. It allows the system to monitor and display the user's heart rate.
3. **LCD Screen (LM044L):** The LCD screen, specifically the LM044L model, is a liquid crystal display that is used to display the heart rate data and other information to the user.
4. **Connecting Wires**: Connecting wires are used to establish electrical connections between the various components.
5. **Ground:** Ground, often represented by the symbol "GND," serves as a common reference point for the electrical signals in the circuit
6. **Switch:** The push button used in this project is used to start measuring the heartbeat.
7. **Potentiometer:** The potentiometer is a variable resistor that can be used to can the value output by the heat beat sensor.
8. **Power Supply:** The power supply 5V ensures that the Arduino, sensors, and LCD screen receive the required voltage and current to function properly.

Chapter 3 Simulation Details

Circuit Design: The circuit was designed and assembled in Proteus, integrating the Arduino Uno, pulse sensor, and LCD screen. Potentiometers were added to control the heart rate signal during the simulation. There were not any default libraries for Arduino and heart rate sensors. We downloaded, imported and set them up in proteus for our project and to design the circuit.

The V_{cc} and GND terminal of potentiometer, heartbeat sensor and Arduino Uno are connected to 5V power and ground respectively. The output of potentiometer is connected to the heartbeat sensor so that when sliding the value of potentiometer results the change in voltage, so heartbeat sensor gives different heartbeat rate. The push button is added to digital pin 2 to start measuring the heartbeat during simulation. Also, the output from heartbeat sensors were connected to digital input pin of Arduino. The outputs were connected to the LCD screen, through the digital pin of Arduino, to analyze and observe the output of heartbeat sensor.

Programming: The Arduino module was programmed using Arduino IDE and compiled there. After compiling, the HEX code is generated, and that path of HEX code is entered in proteus. The hex files of heart rate sensor were downloaded from internet. Arduino is programmed to read data from the pulse sensor, process it, and display the heart rate on the LCD screen. Heart rate sensor is programmed to measure the heart rate by changing the value of potentiometer. After all the programming is done, it is simulated in proteus, and output is observed.

Simulation Setup: We mimic the actual circuit setup; the simulation was started to test whether the functionality and accuracy of heart rate monitoring system is proper or not. During simulation, the LCD screen gives the output of heart rate in digital signal based on sensor input. After running the simulation, we need to press the push button to start the measure of heart rate. We needed to wait for six seconds as Arduino began to count it. During the running phase, we can see the number of high

pulses detected in the screen. To find heartbeat per minute, we multiply the result obtained and multiply it by 10. In LCD Screen we can see how much heartbeat differ were measured when we change the value of resistance in potentiometer.

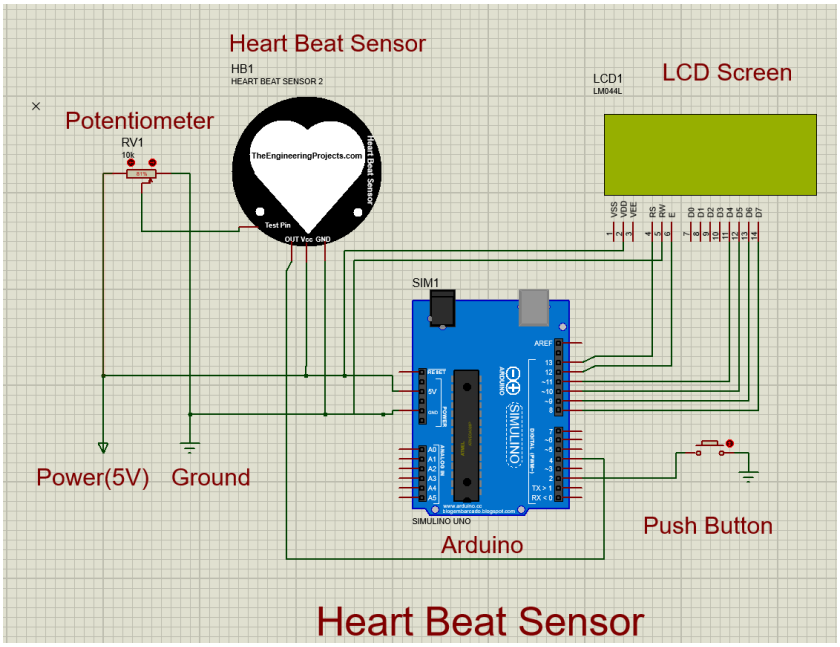

Figure 1.2.1 Heartbeat sensor before simulation

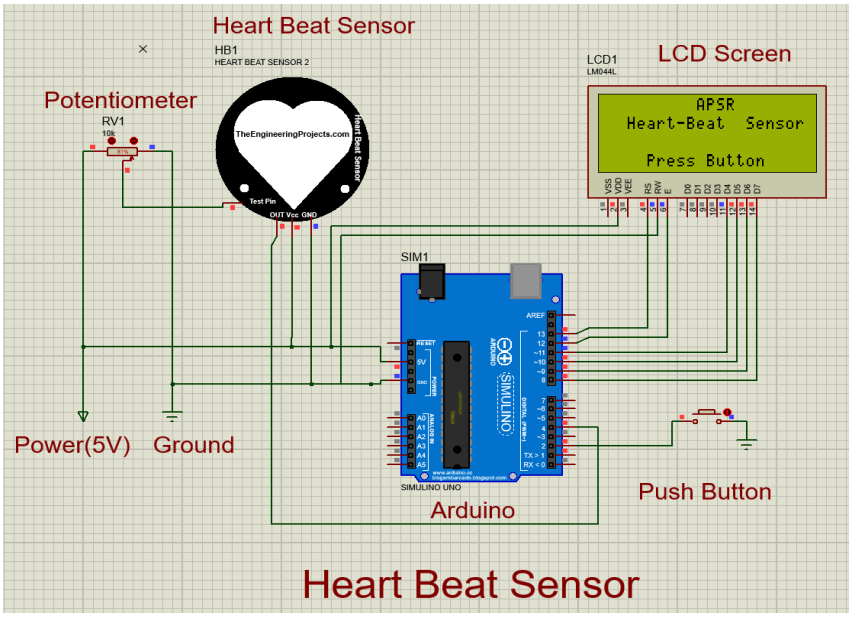

Figure 1.2.2 Heartbeat sensor during simulation when push button is not pressed

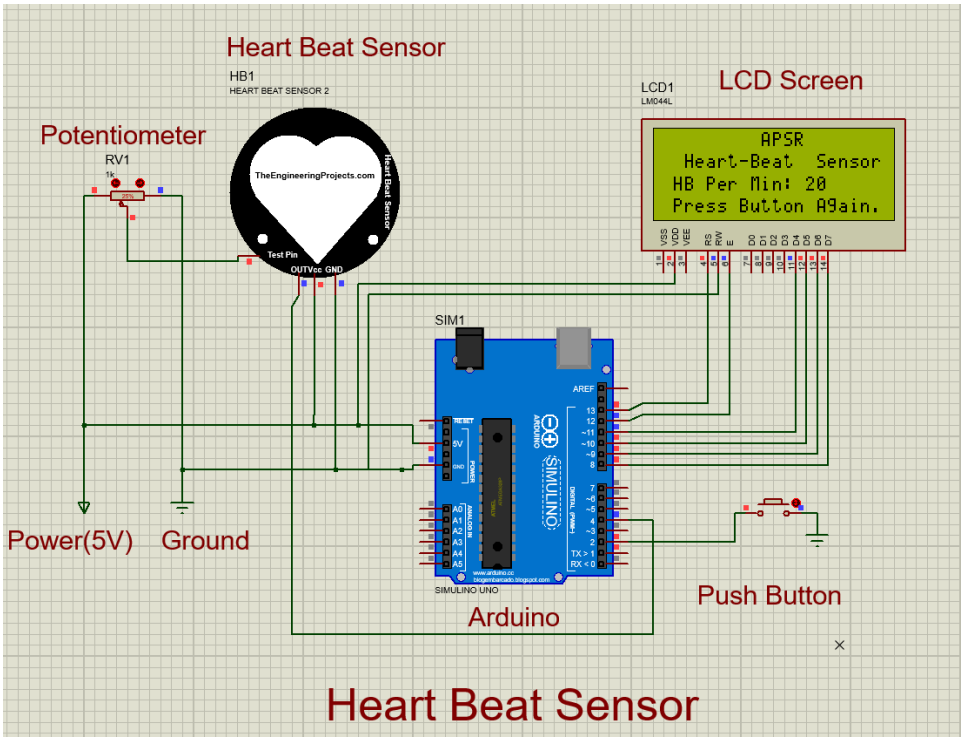

Figure 1.2.3 Heartbeat sensor during simulation (Potentiometer 25%)

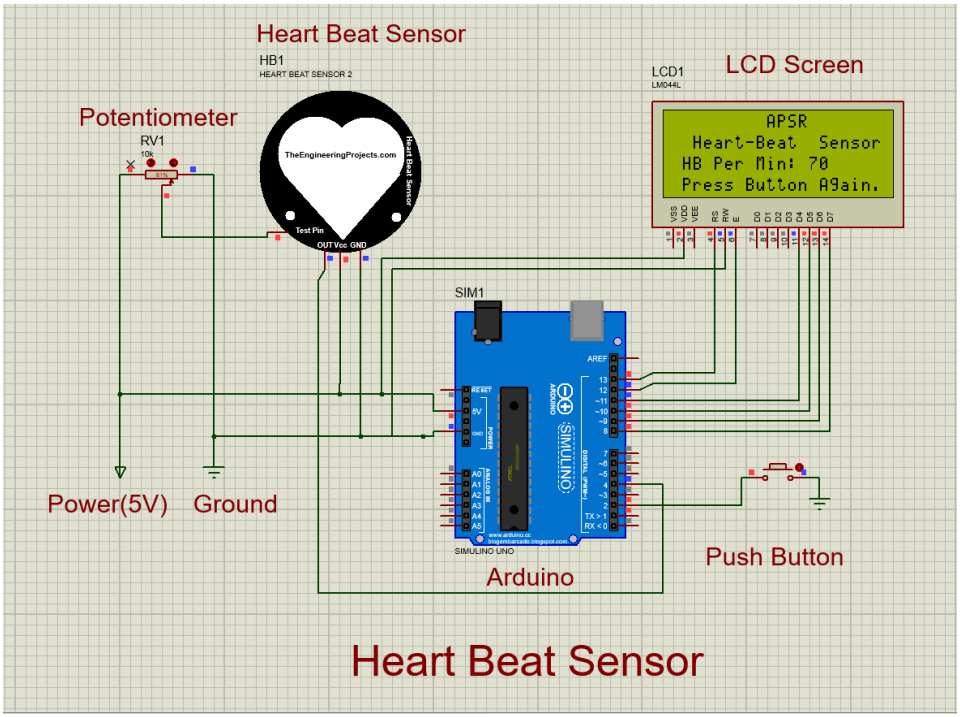

Figure 1.2.4 Heartbeat sensor during simulation (Potentiometer 81%)

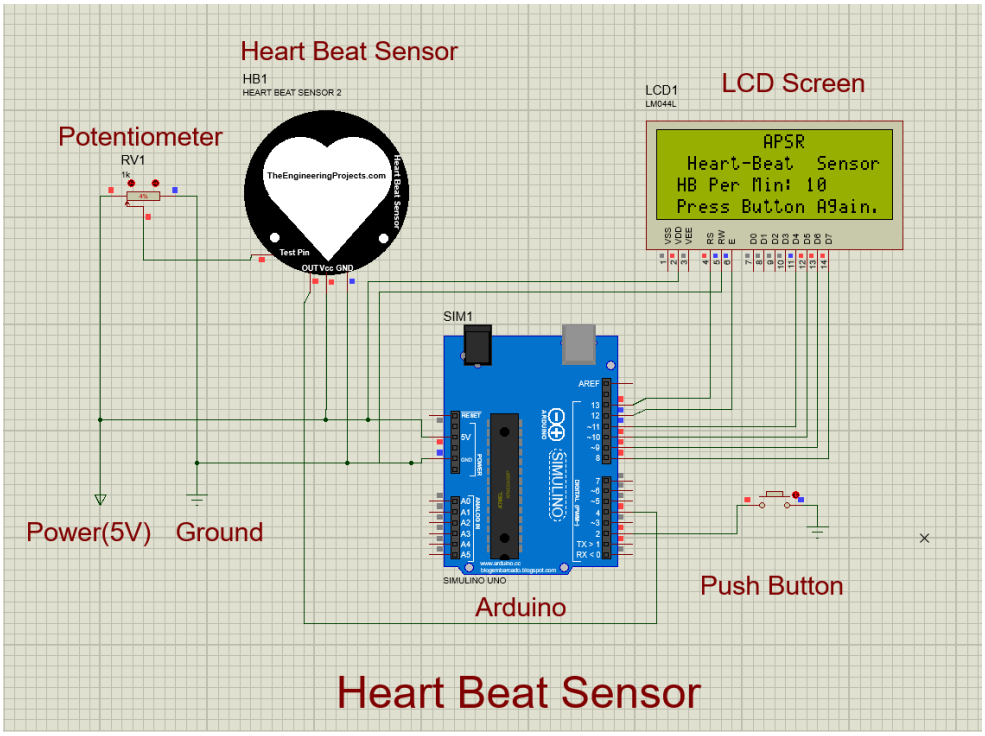

Figure 1.2.5 Heartbeat sensor during simulation (Potentiometer 4%)

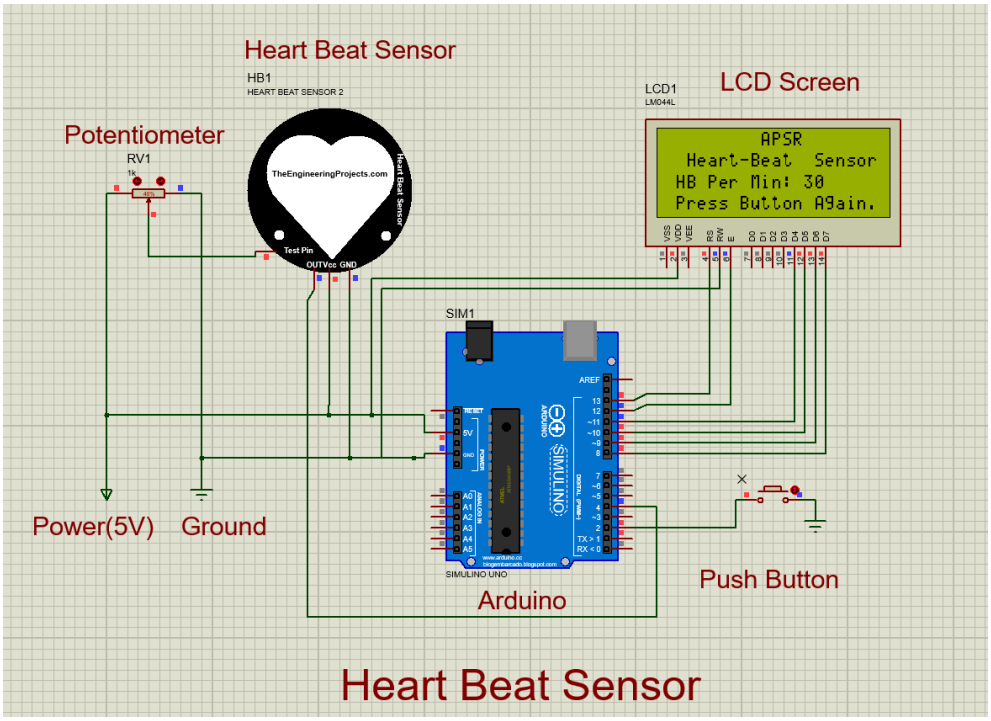

Figure 1.2.6 Heartbeat sensor during simulation (Potentiometer 48%)

Chapter 4 Functional Testing

To ensure the reliability and accuracy of the heart rate monitoring system, various scenarios and data inputs were tested within the simulation using Proteus, Arduino Uno, and an LCD for output.

- **Change in value in potentiometer:**
  For the different value of the heartbeat sensor, the value of the potentiometer was changed. The Arduino Uno processed the signal and displayed the heart rate on the LCD, which accurately reflected the simulated heart rate. As we decrease the value of the potentiometer, the value of the heart rate measure decreases.
- **Change in value of measuring time**
  A different time frame is taken to measure the heart rate. We have taken 6sec and 60 sec for testing purposes. After testing we have found that results in a longer time provide more accurate results.
- **Change in resistance of potentiometer.**
  We have tested the result of the potentiometer when the resistance is at 1k ohm and 10k ohm. The result was relatively like the case where the resistance was 1K ohm.
- **Reversing the terminal of Potentiometer.**
  We have reversed the terminal of Potentiometer, change previous power terminal to ground and ground terminal to power. The condition where heart beat increased is the reverse of the increasing condition of original terminal connection.

Chapter 5 Results and Analysis:

The simulation was run many times and observations were made to obtain following data of Heart Beats per Minute for different resistance levels, observation time and potentiometer value:

Table 1 Heat beat per minute under different scenarios.

Value of potentiometer (%)	Heart Beats per Minute (HBM)			
	Resistance of Potentiometer=1k			Resistance of Potentiometer = 10k
	Time = 6 sec		Time = 60 sec	Time=6 sec
	Normal Condition	Reverse Condition	Normal Condition	Normal Condition
4	10	80	16	20
25	20	60	20	20
48	30	30	28	20
81	70	20	76	70

The analysis of the heart rate monitor data based on variations in potentiometer value, resistance, and observation time show interesting patterns. With a 1k ohm resistance in the normal condition, higher percentage of potentiometer enhances increased heart beats per minute (HBM) throughout all observation times indicative of positive correlation. Conversely, this behavior is reversed under the same resistance setting: a decrease in HBM is observed for lower potentiometer values with an effect of polarity reversal while an increase in potentiometer value correlates with decreased HBM. The sensor reacts to these adjustments through different behavior which contrasts sharply with each other. It has been revealed how sensitive the

potentiometer adjustment is and how much it affects polarity reversal of heart rates on sensors.

Moreover, the observations in different time intervals for normal conditions with 1k ohm resistance, however, indicate a similar pattern; it is in these cases that sixty-second intervals would be more useful. Nevertheless, there is a reduced sensitivity of the sensor when resistance increases to ten thousand ohms under normal conditions and six seconds of observation. Except for an increase at maximum potentiometer setting, very minimal fluctuations occur in HBM even after adjusting on potentiometer. The reduced response to higher resistance implies that heart rate monitoring has had to balance between stability and speed.

Chapter 6 Code for Arduino

```
#include <TimerOne.h>

#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);


const int heartBeatSensorPin = 4; // Sensor Pin

const int startButtonPin = 2;  // Pushbutton Pin


volatile int heartBeatCount = 0;

volatile int elapsedSeconds = 0;

int heartBeatsPerMinute = 0;

bool isCounting = false;

bool isPulseHigh = false;


void setup() {

  pinMode(heartBeatSensorPin, INPUT);

  pinMode(startButtonPin, INPUT_PULLUP);


  Timer1.initialize(1000000); // 1 second interval

  Timer1.attachInterrupt(timerIsr);
```

```
  lcd.begin(20, 4); // Set up the LCD's number of columns
and rows
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("         APSR    ");
  lcd.setCursor(0, 1);
  lcd.print("  Heart-Beat  Sensor  ");
  lcd.setCursor(0, 3);
  lcd.print("    Press Button");

  Serial.begin(9600);
}

void loop() {
  if (digitalRead(startButtonPin) == LOW && !isCounting)
{
    delay(50); // Debounce delay
    if (digitalRead(startButtonPin) == LOW) {
      lcd.setCursor(0, 2);
      lcd.print(" HB Counting... ");
```

```
      lcd.setCursor(0, 3);
      lcd.print(" Please Wait: ");
      isCounting = true;
    }
  }

  if (isCounting) {
    if (digitalRead(heartBeatSensorPin) == HIGH && !isPulseHigh) {
      heartBeatCount++;
      isPulseHigh = true;
      lcd.setCursor(0, 2);
      lcd.print(" High Pulse ");
      lcd.print(heartBeatCount);
    } else if (digitalRead(heartBeatSensorPin) == LOW && isPulseHigh) {
      isPulseHigh = false;
    }
  }
```

```
  if (elapsedSeconds == 6) { // Measure over 6 seconds for better accuracy
    heartBeatsPerMinute = heartBeatCount*10 ; // Convert 6-second count to beats per minute
    isCounting = false;

    lcd.setCursor(0, 2);
    lcd.print(" HB Per Min: ");
    lcd.print(heartBeatsPerMinute);
    lcd.print("  ");
    lcd.setCursor(0, 3);
    lcd.print(" Press Button Again.");

    heartBeatCount = 0;
    elapsedSeconds = 0;

    Serial.print("HB Per Min: ");
    Serial.println(heartBeatsPerMinute);
  }
}
```

```
void timerIsr() {
  if (isCounting) {
    elapsedSeconds++;
    lcd.setCursor(14, 3);
    lcd.print(elapsedSeconds);
    lcd.print("       ");
  }
}
```

Chapter 7 Conclusion and Recommendation

This project successfully demonstrates the development and implementation of a heart rate monitoring system using Proteus and an Arduino microcontroller for simulation The system effectively integrates hardware and software components to provide accurate and reliable heart rate measurements, which are displayed on an LCD screen. The use of a pulse sensor, Arduino microcontroller, and Proteus software has enabled the creation of a cost-effective, flexible, and scalable heart rate monitoring solution.

7.1 Future Enhancement

- **Physical Prototype Development:** Assemble actual hardware components for a physical prototype.
- **Field Testing and Calibration:** Conduct extensive field testing to ensure accuracy and reliability in real-world conditions.
- **Integration with Other Health Metrics:** Expand the system to monitor additional health metrics.
- **Wireless Connectivity:** Add features like Bluetooth or Wi-Fi to sync data with smartphones or cloud-based platforms.

Contribution

Table 2 Contribution table

Name	Contribution
Abiral Adhikari (02)	Coding for the Arduino
Prashant Manandhar (30)	Testing and debugging in various input values.
Samir Wagle (60)	Circuit Design, Simulation and Analysis
Reewaj Khanal (61)	Circuit Design, Simulation and Analysis

GitHub Link

The source code and the proteus file of this project is available at GitHub link: https://github.com/Eemayas/Heart-Beat-Simulation-Proteus