

# **IOT BASED HEALTH MONITORING SYSTEM**

## **INTERDISCIPLINARY PROJECT**

Submitted in partial fulfillment of the requirements for the award of  
Bachelor of Technology degree in Information Technology

by

**ELVIS N (41120059)**



**DEPARTMENT OF INFORMATION TECHNOLOGY  
SCHOOL OF COMPUTING**

# **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY  
(DEEMED TO BE UNIVERSITY)**

**CATEGORY-1 UNIVERSITY BY UGC**

**Accredited with Grade “A++” by NAAC**

**12B Status by UGC |Approved by AICTE**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

**MAY 2024**



# SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

## DEPARTMENT OF INFORMATION TECHNOLOGY

### BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ELVIS N (41120059)** who carried out the project entitled "**IOT BASED HEALTH MONITORING SYSTEM**" under my supervision from January 2024 to May 2024.

Internal Guide

Ms. MERLIN MARY JENITHA, M.Tech.,

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 08.05.2024

Internal Examiner

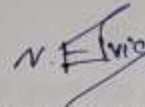
External Examiner

## DECLARATION

I, **ELVIS N (41120059)** hereby declare that the Project Report entitled **IOT BASED HEALTH MONITORING SYSTEM** done by me under the guidance of **Ms.MERLIN MARY JENITHA, M.Tech.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

**DATE:** 08.05.2024

**PLACE:** CHENNAI



**SIGNATURE OF THE CANDIDATE**

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T. Sasikala M.E., Ph. D**, Dean, School of Computing **Dr. R. Subhashini M.E., Ph.D.**, Head of the Department of Information technology for providing me necessary support and details at the right time during the progressive

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. Merlin Mary Jenitha , M.Tech.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

# CERTIFICATE OF COMPLETION



HACKWIT TECHNOLOGIES PVT LTD  
Redefine by Technologies

149, 1C, 2nd Floor, Mount Poonamallee Rd,  
Ramapuram, Chennai - 600089.  
admin@hackwittechnologies.com  
www.hackwittechnologies.com

10.02.2024

## CERTIFICATE FOR INTERNSHIP

TO WHOMSOEVER IT MAY CONCERN

This is to Certify that **ELVIS N (Reg No: 41120059)** student of Sathyabama Institute of Science and Technology, B. Tech Information Technology, has successfully completed the Internship at Hackwit Technologies from 22<sup>nd</sup> January to 30<sup>th</sup> January 2024.

During the internship, **ELVIS N** has closely worked as a part of the **Internet of Things** by carrying out the project entitled "**IOT Based Health Monitoring System**".

He demonstrated good developing and learning skill with self-motivated attitude to learn new things. His contribution to implement the Internet of Things was good. We wish him all the best for his future endeavors.

Best Regards,

Ravikumar S  
CEO & Managing Director  
Hackwit Technologies Pvt Ltd



## CERTIFICATE OF COMPLETION



## **ABSTRACT**

In recent years, the emergence of Internet of Things (IoT) technology has revolutionized various industries, including healthcare. IoT-based health monitoring systems offer significant advancements in remote patient monitoring, allowing for continuous and real-time health data collection, analysis, and intervention. This paper presents the design and implementation of an IoT-based health monitoring system aimed at enhancing the quality of healthcare delivery. The proposed system consists of interconnected wearable sensors and data processing units. Wearable sensors, equipped with various physiological and environmental sensors, collect real-time health data such as heart rate, blood pressure, temperature, and physical activity. These sensors transmit the collected data wirelessly to a data processing unit, typically a smartphone or a dedicated gateway device. The data processing unit preprocesses the raw data and transmits it securely to the cloud-based platform using Internet connectivity. The analytics engine identifies patterns, anomalies, and trends in the data, providing valuable insights into the patient's health status. Furthermore, the system incorporates rule-based logic and personalized health profiles to generate alerts and notifications for healthcare providers and patients in case of any abnormality or emergency situation. The system promotes proactive healthcare management by empowering patients with access to their own health data and encouraging self-care practices.

## TABLE OF CONTENTS

| CHAPTER NO | TITLE                             | PAGE NO |
|------------|-----------------------------------|---------|
|            | ABSTRACT                          | vii     |
|            | LIST OF FIGURES                   | x       |
|            | LIST OF ABBREVIATIONS             | xi      |
| 1          | INTRODUCTION                      | 1       |
|            | 1.1 IOT                           | 2       |
|            | 1.2 PROBLEM STATEMENT             | 2       |
|            | 1.3 OBJECTIVES                    | 3       |
| 2          | METHODOLOGY                       | 4       |
|            | 2.1 AIM OF THE PROJECT            | 4       |
|            | 2.2 SCOPE OF THE PROJECT          | 4       |
|            | 2.3 HARDWARE/SOFTWARE REQUIREMENT | 5       |
|            | 2.3.1 WINDOWS/MAC/LINUX SYSTEM    | 6       |
|            | 2.3.2 ARDUINO IDE                 | 6       |
|            | 2.3.3 ARDUINO UNO                 | 6       |
|            | 2.3.4 PULSE SENSOR                | 7       |
|            | 2.3.5 TEMPERATURE SENSOR          | 7       |
|            | 2.3.6 BREADBOARD                  | 7       |
|            | 2.3.7 RESISTORS                   | 8       |
|            | 2.3.8 LCD                         | 8       |
|            | 2.3.9 LED                         | 9       |
|            | 2.3.10 ESP8266                    | 9       |
|            | 2.3.11 JUMPER WIRES               | 9       |
|            | 2.3.12 BUZZER                     | 10      |
|            | 2.3.13 I2C                        | 10      |
|            | 2.4 LIBRARIES USED                | 11      |

|          |     |                                   |           |
|----------|-----|-----------------------------------|-----------|
|          | 2.5 | MODULES                           | 12        |
|          | 2.6 | IMPLEMENTATION                    | 14        |
| <b>3</b> |     | <b>RESULTS AND DISCUSSION</b>     | <b>18</b> |
|          | 3.1 | TESTS                             | 18        |
|          | 3.2 | RESULTS                           | 20        |
| <b>4</b> |     | <b>CONCLUSION AND FUTURE WORK</b> | <b>22</b> |
|          | 4.1 | CONCLUSIONS                       | 22        |
|          | 4.2 | FUTURE WORK                       | 22        |
|          |     | <b>REFERENCES</b>                 | <b>25</b> |
|          |     | <b>APPENDICES</b>                 | <b>26</b> |
|          | A.  | SOURCE CODE                       | 26        |
|          | B.  | SCREENSHOTS                       | 29        |

## LIST OF FIGURES

| FIGURE NO | NAME               | PAGE NO |
|-----------|--------------------|---------|
| 2.1       | ARDUINO UNO        | 6       |
| 2.2       | PULSE SENSOR       | 7       |
| 2.3       | TEMPERATURE SENSOR | 7       |
| 2.4       | BREADBOARD         | 8       |
| 2.5       | RESISTOR           | 8       |
| 2.6       | LCD                | 8       |
| 2.7       | LED                | 9       |
| 2.8       | ESP8266            | 9       |
| 2.9       | JUMPER WIRES       | 10      |
| 2.10      | BUZZER             | 10      |
| 2.11      | I2C                | 11      |
| 2.12      | CIRCUIT DIAGRAM    | 13      |
| 2.13      | SETUP FUNCTION     | 16      |
| 2.14      | LOOP FUNCTION      | 17      |

## LIST OF ABBREVIATIONS

| ABBREVIATION | EXPANSION                          |
|--------------|------------------------------------|
| LCD          | Liquid Crystal Display             |
| IDE          | Integrated Development Environment |
| ESP          | Event Stream Processing            |
| MQTT         | Message Query Telemetry Transport  |
| IOT          | Internet Of Things                 |
| TLS          | Transport Layer Security           |
| EMR          | Electronic Medical Records         |
| API          | Application Programming Interface  |
| SSL          | Secure Socket Layer                |
| LED          | Light Emitting Diode               |
| I2C          | Inter Integrated Circuit           |

# **CHAPTER 1**

## **INTRODUCTION**

The integration of IOT technology in healthcare has led to transformative advancements in remote patient monitoring and healthcare management. This introduction provides an overview of the IoT-based health monitoring system, highlighting its significance in improving healthcare delivery and enhancing patient outcomes.

The healthcare industry is facing numerous challenges, including an aging population, rising chronic diseases, and increasing healthcare costs. Traditional healthcare delivery models often struggle to provide timely and personalized care, particularly for patients with chronic conditions who require continuous monitoring and intervention. In this context, IOT technology offers a promising solution by enabling the development of innovative health monitoring systems that leverage interconnected devices and real-time data analytics.

An IOT based health monitoring system typically comprises wearable sensors, data processing units, and a cloud-based platform. These components work together to collect, transmit, and analyze health data, facilitating remote monitoring and timely intervention. Wearable sensors, equipped with physiological and environmental sensors, gather various health parameters such as heart rate, blood pressure, temperature, and physical activity. These sensors transmit the collected data wirelessly to a data processing unit, which preprocesses the data and transmits it securely to the cloud-based platform.

At the heart of the system lies a robust data analytics engine, which employs machine learning algorithms and statistical techniques to analyze the incoming health data. The analytics engine identifies patterns, anomalies, and trends in the data, providing valuable insights into the patient's health status. Moreover, the system incorporates rule-based logic and personalized health profiles to generate alerts and notifications for healthcare providers and patients in case of any abnormality or emergency situation.

The IOT based health monitoring system offers several key advantages over traditional healthcare monitoring methods. Firstly, it enables continuous and remote monitoring of patients, allowing healthcare providers to track their health status in real-time without the need for frequent hospital visits. This not only improves the efficiency of healthcare delivery but also enhances patient comfort and convenience. Secondly, the system facilitates early detection of health issues by analyzing subtle changes in health parameters over time, enabling timely intervention and preventive care. Lastly, the system empowers patients by providing them with access to their own health data and encouraging self-care practices.

In summary, the integration of IOT technology in healthcare has the potential to revolutionize healthcare delivery by enabling continuous, remote, and proactive health monitoring. By leveraging interconnected devices and real-time data analytics, IOT based health monitoring systems have the capacity to improve patient outcomes, reduce healthcare costs, and enhance overall quality of life.

## **1.1 IOT**

It refers to the network of interconnected devices that can communicate and exchange data with each other over the internet without requiring human intervention. These devices can range from everyday objects like household appliances, wearable devices, and industrial machines to vehicles and infrastructure components. IoT enables these devices to collect, analyze, and transmit data, leading to improved efficiency, automation, and decision-making in various domains such as healthcare, transportation, agriculture, and smart cities.

## **1.2 Problem Statement**

In the absence of the doctors, the patient cannot consult the doctors due to which emergency situation may also be created. Personal health monitoring is crucial due to rising health problems exacerbated by stressful lifestyles. Stressful lifestyles are significantly impacting public health. Overcrowded hospitals and high doctor fees disproportionately affect those unable to afford consultations. Soaring doctor fees disproportionately affect those unable to afford consultations. Soaring doctor fees are impacting patients, particularly those unable to afford

consultations. Patients often discover minor ailments only after paying substantial fees for consultations.

### **1.3 Objectives**

To Enable continuous monitoring of vital signs and health parameters such as heart rate, blood pressure, temperature, etc., without the need for frequent hospital visits .To Detect early signs of health deterioration or potential medical emergencies through real-time data analysis, allowing for timely intervention and treatment .To Facilitate remote patient monitoring, especially for individuals with chronic illnesses or those who require regular medical attention, enabling healthcare providers to remotely assess their condition and adjust treatment plans as needed.To Provide personalized healthcare insights and recommendations based on individual health data collected over time, tailored to each person's unique health profile and needs.To Reduce healthcare costs associated with hospital visits, emergency room admissions, and unnecessary medical procedures by shifting towards preventive and remote healthcare monitoring models.

## CHAPTER 2

### METHODOLOGY

#### 2.1 AIM OF THE PROJECT

The aim of the investigation is to design, develop, and evaluate an IoT-based health monitoring system that enables continuous, remote, and proactive monitoring of patients' health status. The system aims to leverage interconnected devices, real time data analytics, and remote communication technologies to enhance healthcare delivery, improve patient outcomes, and promote preventive care.

#### 2.2 SCOPE OF THE PROJECT

Scope of IoT-Based Health Monitoring System:

- **Hardware and Software Development:** The scope involves the design, development, and integration of hardware components (wearable sensors, data processing units) and software components (cloud-based platform, data analytics engine) required for the IoT-based health monitoring system.
- **Data Collection and Transmission:** This includes defining protocols for data collection from wearable sensors, ensuring compatibility and seamless integration of different sensor types, and establishing secure and efficient data transmission mechanisms to the cloud-based platform.
- **Data Preprocessing and Analytics:** The system will preprocess the raw health data to clean noise, aggregate data, and format it for analysis. Implementing machine learning algorithms and statistical techniques for data analytics to derive insights into patients' health status is also within scope.
- **Alert Generation and Notification:** Defining rules and thresholds for identifying abnormal health conditions, generating alerts, and notifying healthcare providers and patients through various channels such as email, SMS, or dedicated dashboard interfaces falls under scope.
- **Security and Privacy:** Ensuring data security and privacy throughout the system, including data encryption during transmission, access control mechanisms, and compliance with relevant regulations such as HIPAA (Health Insurance Portability and Accountability Act) or GDPR (General Data Protection Regulation).
- **User Interface and Experience:** Designing intuitive user interfaces for both

healthcare providers and patients to access and interact with the system, visualize health data, configure alerts, and manage preferences.

- **Testing and Validation:** Conducting thorough testing and validation of the system to ensure its accuracy, reliability, scalability, and usability in real-world healthcare settings. This includes simulated testing as well as trials with actual users.
- **Scalability and Adaptability:** Designing the system to be scalable and adaptable to accommodate varying numbers of users, sensor types, and healthcare environments. Considering future technological advancements and potential integration with other healthcare systems or devices is also within scope.
- **Regulatory Compliance:** Ensuring compliance with relevant regulatory requirements and standards governing healthcare data management, privacy, and security.
- **Potential Applications:** Exploring potential applications of the IoT-based health monitoring system in various healthcare domains such as chronic disease management, elderly care, post-operative monitoring, and wellness management.

Overall, the scope of the IoT-based health monitoring system encompasses hardware and software development, data collection and analysis, alert generation and notification, security and privacy considerations, user interface design, testing and validation, scalability, regulatory compliance, and potential applications in healthcare.

## 2.3 HARDWARE/SOFTWARE REQUIREMENT

- Windows/Mac/Linux Based System
- Arduino IDE
- Arduino UNO
- Pulse Sensor
- Temperature Sensor
- Breadboard
- Resistors
- LCD
- LED
- ESP8266

- Jumper Wires

### **2.3.1 Windows/Mac/Linux System**

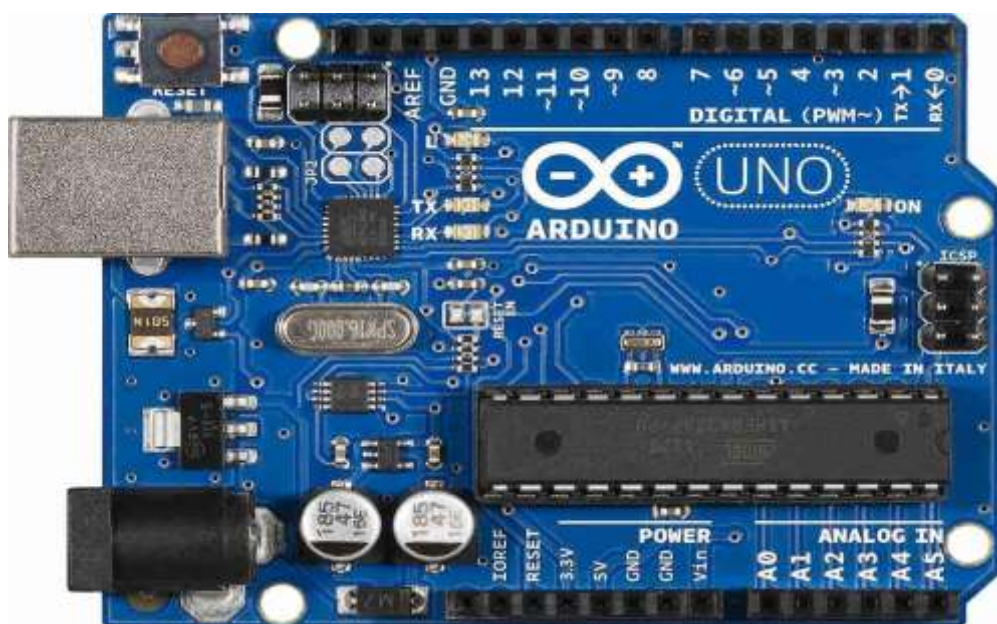
Windows, Mac or Linux based operating system for personal computers or any other internet of things devices. The system requirements for the Arduino IDE are relatively minimal, making it compatible with various operating systems.

### **2.3.2 Arduino IDE**

The Arduino software, also known as the Arduino IDE (Integrated Development Environment), is a platform used for writing, compiling, and uploading code to Arduino microcontroller-based boards. The Arduino software provides a user-friendly and accessible platform for programming Arduino boards, making it suitable for beginners and experienced users alike to create a wide range of embedded projects.

### **2.3.3 Arduino UNO**

The Arduino Uno is one of the most popular and widely used microcontroller boards in the Arduino ecosystem. It is based on the ATmega328P microcontroller and offers a wide range of features and capabilities for prototyping and building embedded projects.



**Fig: 2.1 ARDUINO UNO**

#### **2.3.4 Pulse Sensor**

A pulse sensor is a type of sensor designed to measure the heartbeat rate or pulse of an individual. It typically works by detecting changes in blood volume in the microvascular bed of tissue under the sensor. This change in blood volume occurs with each heartbeat, causing fluctuations in light absorption or reflection, depending on the sensor's configuration.



***Fig: 2.2 PULSE SENSOR***

#### **2.3.5 Temperature Sensor**

When selecting a temperature sensor for patient monitoring, factors such as accuracy, response time, comfort, and hygiene should be considered. Additionally, the sensor's compatibility with the monitoring system and its ease of integration should also be taken into account. Ultimately, the choice of temperature sensor will depend on the specific requirements of the monitoring application and the preferences of healthcare professionals.



***Fig: 2.3 TEMPERATURE SENSOR***

#### **2.3.6 Breadboard**

A breadboard is a versatile and widely used tool in electronics prototyping and circuit design. It allows electronic components to be easily connected and tested without the need for soldering. Breadboards are essential tools for electronics hobbyists, students, and professionals alike, providing a convenient platform for experimenting

with circuit designs and quickly bringing ideas to life.



***Fig: 2.4 BREADBOARD***

### **2.3.7 Resistors**

Resistors are passive electronic components that impede the flow of electric current. They are widely used in electronic circuits to control the amount of current flowing through various parts of the circuit, adjust signal levels, and divide voltages.



***Fig: 2.5 RESISTOR***

### **2.3.8 LCD**

LCD (Liquid Crystal Display) modules compatible with Arduino are commonly used for displaying text, numbers, and simple graphics in Arduino projects. These LCD modules come in various sizes and configurations, but one of the most popular choices is the 16x2 LCD module, which can display 16 characters per line and has 2 lines.



***Fig: 2.6 LCD***

### **2.3.9 LED**

LED lights are a type of lighting that has gained immense popularity in recent years due to their energy efficiency, longevity, and versatility. Unlike traditional incandescent bulbs, which use a filament to produce light, LEDs produce light through the movement of electrons in a semiconductor material.



**Fig: 2.7 LED**

### **2.3.10 ESP8266**

The ESP8266 is a highly popular and versatile Wi-Fi module developed by Espressif Systems. It is widely used in IoT (Internet of Things) and embedded systems projects due to its low cost, low power consumption, and built-in Wi-Fi capabilities.



**Fig: 2.8 ESP8266**

### **2.3.11 Jumper Wires**

Jumper wires are electrical wires with connectors at each end, typically used to create temporary connections between electronic components on a breadboard or between a breadboard and other components. Jumper wires are essential tools for electronics prototyping and experimentation, providing a flexible and convenient way to create temporary connections between components in breadboard-based circuits.



***Fig 2.9 JUMPER WIRES***

### **2.3.12 Buzzer**

A buzzer is an electromechanical device that produces a buzzing or buzzing sound when an electric current passes through it. It typically consists of a vibrating reed or membrane, an electromagnetic coil, and a housing to amplify and project the sound. Buzzers play a crucial role in alerting users to important events and conditions in electronic devices and systems, contributing to their usability and safety.



***Fig 2.10 BUZZER***

### **2.2.13 I2C**

LED displays controlled via I2C typically consist of a matrix of LEDs arranged in rows and columns. Each LED can be individually controlled by sending commands over the I2C bus to the display controller. This allows for versatile control over the display, including turning individual LEDs on or off, adjusting brightness levels, and displaying characters, symbols, or images.



***Fig 2.11 I2C***

## **2.4 LIBRARIES USED**

For an IOT-based health monitoring system, several libraries can be useful depending on the specific requirements of the project. Here are some common libraries used in developing IoT-based health monitoring systems:

- **MQTT Libraries:** MQTT is a lightweight messaging protocol commonly used in IoT applications for communication between devices and servers. Libraries such as PubSubClient for Arduino or Paho MQTT for Python provide MQTT client implementations that enable devices to publish sensor data and subscribe to commands or updates from the server.
- **WiFi Libraries:** WiFi libraries are essential for connecting IoT devices to a wireless network. For Arduino-based projects, libraries such as ESP8266WiFi or WiFi101 are commonly used to establish WiFi connections and configure network settings.
- **Web Server Libraries:** Web server libraries enable IoT devices to host web pages or APIs for remote monitoring and control. Libraries such as ESP8266WebServer or Flask for Python facilitate the creation of web interfaces where users can view health data, configure settings, or receive alerts.
- **Sensor Libraries:** Depending on the sensors used for health monitoring (e.g., heart rate sensors, temperature sensors), specific sensor libraries may be required to interface with the sensors and collect data. Many sensor manufacturers provide Arduino or Python libraries for their products,

simplifying the integration process.

- **Data Visualization Libraries:** Libraries such as Chart.js, Plotly, or Matplotlib can be used to visualize health data in real-time or generate historical charts and graphs for analysis. These libraries enable users to gain insights from the collected data and track trends over time.
- **Database Libraries:** If data logging or storage is required, database libraries such as SQLite (for local storage) or Firebase Realtime Database (for cloud storage) can be used to store and retrieve health data. These libraries provide functionalities for data management, querying, and synchronization between devices.
- **Security Libraries:** Security libraries such as Crypto Arduino or Python's hash lib module can be used to implement encryption and hashing algorithms for securing sensitive health data during transmission or storage. Additionally, libraries like TLS for Arduino or Python's SSL module provide secure communication over HTTPS.
- **Notification Libraries:** Libraries such as Twilio, Push bullet, or Telegram Bot can be integrated to send notifications or alerts to users' smartphones or other devices based on predefined conditions (e.g., abnormal health readings, medication reminders).

## **2.5 MODULES**

A microcontroller or microprocessor to process the data, and a communication module to transmit the data to a remote server or cloud platform for further analysis and storage. Here's a simplified circuit diagram explanation for such a system:

- **Sensor Module:** This module consists of various sensors to measure different health parameters such as heart rate, body temperature, blood pressure, oxygen saturation, etc. Each sensor is interfaced with the microcontroller through appropriate signal conditioning circuits. For example, a heart rate sensor might use photoplethysmography (PPG) to detect blood volume changes in the microvasculature.
- **Microcontroller or Microprocessor:** The microcontroller acts as the brain of

**Communication Module:** This module enables the IoT capability of the system by allowing it to communicate with external devices or services over the internet. Common communication modules include Wi-Fi, Bluetooth, Zigbee, and GSM/GPRS modules. The microcontroller sends the processed health data to the communication module, which then transmits it to a remote server or cloud platform for storage and analysis.

- ### Circuit Diagram & Connections
- 
- The diagram illustrates the hardware setup for the project. An ESP8266 module is connected to an Arduino Uno via a USB cable. The Arduino Uno is connected to an LCD display showing 'BPM : 82' and 'Temp : 97.43 F'. A breadboard contains an LED, a 1K resistor, and a 2K resistor. A pulse sensor is also connected to the breadboard. The connections are as follows:
- ESP8266 module is connected to the Arduino Uno via a USB cable.
  - Arduino Uno is connected to the LCD display.
  - Breadboard components (LED, 1K resistor, 2K resistor) are connected to the Arduino Uno and the pulse sensor.
  - Pulse sensor is connected to the breadboard.

## **2.6 IMPLEMENTATION**

Implementing a health monitoring system using IoT involves several steps, including hardware setup, data collection, transmission, storage, and visualization. Here's a general overview of how you can implement such a system:

### **1. Hardware Setup:**

- Choose appropriate sensors for monitoring vital signs such as heart rate, body temperature, blood pressure, oxygen saturation, etc.
- Interface these sensors with a microcontroller or development board capable of connecting to the internet, such as Arduino with an ESP8266 or ESP32 module, Raspberry Pi, or other IoT platforms.
- Power the system using a reliable power source, considering the power requirements of the sensors and the microcontroller.

### **2. Data Collection:**

- Program the microcontroller to read data from the sensors at regular intervals.
- Ensure that the sensor readings are accurate and reliable by calibrating the sensors if necessary.
- Handle any error conditions or sensor failures gracefully to prevent erroneous data.

### **3. Data Transmission:**

- Use a communication protocol such as MQTT, HTTP, or CoAP to transmit the sensor data from the microcontroller to a cloud server or gateway.
- Set up a secure connection (e.g., TLS/SSL) to protect sensitive health data during transmission over the internet.
- Implement error handling and retry mechanisms to ensure reliable data transmission, especially in unstable network conditions.

### **4. Data Storage:**

- Choose a suitable database or data storage solution for storing the collected health data.

- Consider factors such as scalability, reliability, and data retention requirements when selecting a storage solution.
- If using a cloud-based storage service, ensure compliance with data privacy regulations such as GDPR or HIPAA.

#### **5. Data Visualization:**

- Develop a web-based dashboard or mobile application to visualize the health data in real-time.
- Use data visualization libraries such as Chart.js, Plotly, or Matplotlib to create interactive charts and graphs for displaying vital signs over time.
- Include features for setting thresholds, generating alerts, and viewing historical data trends to help users monitor their health status effectively.

#### **6. User Authentication and Access Control:**

- Implement user authentication mechanisms to ensure that only authorized users can access the health monitoring system.
- Use strong encryption techniques to protect user credentials and sensitive health data stored in the system.
- Define access control policies to restrict access to certain features or data based on user roles and permissions.

#### **7. Integration with External Services:**

- Integrate the health monitoring system with external services such as healthcare providers, emergency services, or telemedicine platforms to enable remote monitoring, consultation, and emergency response.
- Implement APIs or webhooks for seamless integration with third-party systems and services.

#### **8. Testing and Validation:**

- Thoroughly test the entire system, including hardware components, software modules, data transmission, storage, and visualization features.
- Conduct usability testing with end-users to gather feedback and identify areas for improvement.
- Validate the accuracy, reliability, and security of the system through rigorous testing and validation procedures.

## 9. Deployment and Maintenance:

- Deploy the health monitoring system in the target environment, whether it's a home setting, healthcare facility, or remote location.
- Monitor system performance and reliability continuously and perform regular maintenance tasks such as software updates, security patches, and hardware replacements as needed.
- Provide ongoing technical support and user training to ensure the smooth operation of the system and user satisfaction.

## 10. Setup Method:

- In Arduino programming, the void setup() function is a crucial part of every sketch. It's where you initialize variables, pin modes, and other setup-related tasks before the main loop begins executing
- Initialization: The setup() function is primarily used for initializing variables, setting pin modes (like input or output), and configuring any hardware peripherals.
- First Execution: After the Arduino board is powered up or reset, the microcontroller starts executing the code from the setup() function.



```
void setup()
{
  Wire.begin();
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);
  sensors.begin();
  numberOfDevices = sensors.getDeviceCount();
  pulseSensor.analogInput(PulseWire);
  pulseSensor.setThreshold(Threshold);
  pulseSensor.begin();
  pinMode(buzzer, OUTPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(buzzer, LOW);
  lcd.setCursor(0, 0);
  lcd.print("Patient Health");
  lcd.setCursor(0, 1);
  lcd.print("Monitoring System");
  delay(1500);
  lcd.clear();
  calibrateSensor();
}
```

### 2.13 SETUP FUNCTION

## 11. Loop Method

- In Arduino programming, the loop() function is where the main program logic resides. It's executed continuously after the setup() function completes its initialization tasks.
- Continuous Execution: The code inside the loop() function is executed repeatedly in a loop as long as the Arduino is powered up or until it's reset.
- Reading sensors: Polling sensor data and reacting accordingly.
- Controlling outputs: Changing the state of LEDs, motors, or other actuators.
- Responding to events: Handling user inputs, communication events, or external triggers.



```
health_monitoring(Arduino IDE)
File Edit Sketch Tools Help

health_monitoring

void loop()
{
  myHRM = pulseSensor.getHeartRate();
  if (pulseSensor.readStartOfBeat())
  {
    beatEffect(); // Turn on both buzzer and LED with rhythm
    led.write(255, 0);
    led.write(0, 255);
    led.print("Heart:");
    led.print(myHRM * BPMcalibration); // Apply calibration offset
    led.print(" BPM");
    Serial.print("Heart Rate: ");
    Serial.print(myHRM * BPMcalibration); // Apply calibration offset
    Serial.println(" BPM");
    // Print temperature to PulseSensor to serial monitor
    float tempF = readTemperature();
    Serial.print("Temperature: ");
    Serial.print(tempF);
    Serial.println(" F");
    led.write(255, 255);
    led.print("Temp:");
    led.print(tempF);
    led.print(" F");
  }
  else
  {
    digitalWrite(buzzer, LOW); // Ensure buzzer is off if no beat detected
    digitalWrite(ledPin, LOW); // Ensure LED is off if no beat detected
  }
}
```

**FIG 2.14 LOOP FUNCTION**

## **CHAPTER 3**

### **RESULTS AND DISCUSSION**

#### **3.1 TESTS**

Testing a health monitoring system using IoT involves verifying various aspects of the system, including hardware functionality, sensor accuracy, data transmission reliability, and user interface usability. Here's a suggested approach for testing such a system:

##### **1. Hardware Testing:**

- Ensure that all hardware components, including sensors, microcontrollers, communication modules, and power supplies, are functioning correctly.
- Test each sensor individually to verify its accuracy and reliability in measuring vital signs such as heart rate, body temperature, blood pressure, and oxygen saturation.
- Verify that the hardware setup is properly configured and wired according to the system design specifications.

##### **2. Communication Testing:**

- Test the communication between the IoT device and the cloud server or gateway to ensure reliable data transmission.
- Verify that data packets are sent and received correctly without loss or corruption.
- Test the system's resilience to network disruptions, packet loss, and latency by simulating various network conditions.

##### **3. Data Validation:**

- Validate the accuracy of the sensor data by comparing it against reference measurements obtained from certified medical devices or clinical standards.
- Perform data integrity checks to ensure that sensor readings are within acceptable ranges and free from errors or anomalies

##### **4. Functional Testing:**

- Test the functionality of the system features, such as real-time monitoring, data visualization, alerting, and user interaction.

- Verify that users can access the system interface, view health data, set thresholds, and receive alerts as intended.
- Test edge cases and error scenarios to ensure that the system behaves predictably and gracefully handles exceptions.

#### **5. Security Testing:**

- Perform security testing to identify and mitigate vulnerabilities in the system, including unauthorized access, data breaches, and privacy risks.
- Test the strength of encryption mechanisms used to secure data transmission and storage.
- Verify that user authentication, access control, and encryption protocols are implemented correctly and effectively.

#### **6. Usability Testing:**

- Conduct usability testing with end-users to evaluate the user experience and interface design.
- Gather feedback on the ease of use, clarity of information, responsiveness, and overall satisfaction with the system.
- Use surveys, interviews, or user observation sessions to identify usability issues and areas for improvement.

#### **7. Performance Testing:**

- Measure the performance of the system in terms of data throughput, response time, and resource utilization.
- Test the scalability of the system by simulating a large number of concurrent users or devices.
- Identify any bottlenecks or performance limitations and optimize the system architecture and algorithms accordingly.

#### **8. Integration Testing:**

- Test the integration of the health monitoring system with external services, such as healthcare providers, electronic medical records (EMR) systems, or third-party applications.
- Verify that data exchange, interoperability, and compatibility with external systems are seamless and reliable.

#### **9. Regression Testing:**

- Perform regression testing to ensure that new updates or changes to

the system do not introduce regressions or unintended side effects.

- Re-run previous test cases and verify that existing functionality remains intact after modifications to the system.

#### 10. Documentation and Reporting:

- Document test cases, test results, and any issues or defects discovered during testing.
- Prepare a comprehensive test report summarizing the testing process, outcomes, and recommendations for improvement.

### 3.2 RESULTS

The results of a health monitoring system using IoT can vary depending on various factors, including the design of the system, the accuracy of the sensors, the reliability of the communication network, and the effectiveness of data processing and analysis algorithms. Here are some potential results and outcomes of implementing such a system:

- **Early Detection of Health Issues:** IoT-based health monitoring systems can facilitate the early detection of health issues by continuously monitoring vital signs such as heart rate, blood pressure, body temperature, and oxygen saturation. Abnormalities or deviations from normal ranges can trigger alerts or notifications to healthcare providers or caregivers, enabling timely intervention and treatment.
- **Improved Chronic Disease Management:** For individuals with chronic conditions such as diabetes, hypertension, or cardiovascular disease, IoT-based health monitoring systems can help monitor key health indicators and track disease progression over time. By providing real-time feedback and personalized insights, these systems empower patients to better manage their health and make informed decisions about their care.
- **Remote Patient Monitoring:** IoT technology enables remote monitoring of patients outside traditional healthcare settings, such as in their homes or communities. This allows healthcare providers to monitor patients' health status remotely, reducing the need for frequent hospital visits and improving access to care, especially for individuals in rural or underserved areas.
- **Enhanced Care Coordination:** By facilitating communication and data

sharing between patients, caregivers, and healthcare providers, IoT-based health monitoring systems support collaborative care coordination and interdisciplinary healthcare teams. This improves care continuity, reduces duplication of services, and ensures that patients receive timely and appropriate care across different care settings.

- **Personalized Healthcare Interventions:** IoT-based health monitoring systems generate vast amounts of health data that can be analysed to identify patterns, trends, and correlations related to individual health status and behaviour. This enables personalized healthcare interventions tailored to the specific needs and preferences of each patient, leading to more effective and efficient healthcare delivery.
- **Reduced Healthcare Costs:** By preventing complications, minimizing hospital readmissions, and optimizing resource utilization, IoT-based health monitoring systems can help reduce healthcare costs and improve the overall efficiency of the healthcare system. Remote monitoring and proactive interventions can lead to better health outcomes and lower healthcare expenditures over time.
- **Empowerment of Patients:** IoT-based health monitoring systems empower patients to take an active role in managing their health and well-being. By providing access to real-time health data, educational resources, and self-management tools, these systems enable patients to make informed decisions, track their progress, and engage in preventive health behaviours.
- **Population Health Management:** Aggregated data from IoT-based health monitoring systems can be used for population health management initiatives, such as disease surveillance, epidemiological research, and public health interventions. By analysing trends and patterns at the population level, healthcare organizations can identify areas for improvement and implement targeted interventions to promote health and prevent disease.

## CHAPTER 4

### CONCLUSION AND FUTURE WORK

#### 4.1 CONCLUSIONS

In summary, the implementation of a health monitoring system using IoT technology offers numerous benefits and opportunities for improving healthcare delivery and patient outcomes. Here are the key points highlighted in the discussion:

1. **Overview:** IoT-based health monitoring systems utilize sensors, microcontrollers, communication networks, and data processing algorithms to collect, transmit, store, and analyze health data in real-time.
2. **Components:** The system consists of hardware components such as sensors for measuring vital signs (e.g., heart rate, body temperature), microcontrollers or development boards for data processing, communication modules for transmitting data over the internet, and software applications for data visualization and analysis.
3. **Functionality:** The system continuously monitors health parameters, detects abnormalities or emergencies, alerts healthcare providers or caregivers, facilitates remote consultation and intervention, and enables personalized healthcare interventions based on individual health data.
4. **Conclusion:** IoT-based health monitoring systems have the potential to revolutionize healthcare by leveraging technology to enable continuous, remote, and personalized monitoring of health parameters.

#### 4.2 FUTURE WORK

Future work in IoT-based health monitoring systems could focus on several key areas to further enhance their effectiveness and impact:

1. **Enhanced Sensor Technology:** Research and development efforts can be directed towards improving the accuracy, reliability, and miniaturization of sensors used in health monitoring devices. This includes advancements in biosensors, wearables, and implantable devices to collect a wider range of physiological data with higher precision.
2. **Data Analytics and AI:** There is a need for more sophisticated data analytics

algorithms and artificial intelligence (AI) models to extract meaningful insights from the vast amount of data generated by IoT devices. AI can help in predictive analytics for early detection of health issues, personalized treatment recommendations, and anomaly detection for proactive interventions.

- 3. Integration with Electronic Health Records (EHR):** Seamless integration of IoT health data with existing EHR systems is essential for providing comprehensive patient care. Future work should focus on standardizing data formats, interoperability protocols, and secure data exchange mechanisms between IoT devices and EHR platforms.
- 4. Privacy and Security:** Addressing privacy concerns and ensuring the security of sensitive health data remains a critical area for future research. This involves implementing robust encryption techniques, access control mechanisms, and compliance with privacy regulations such as GDPR and HIPAA.
- 5. Remote Monitoring of Chronic Conditions:** Expanding the scope of IoT-based health monitoring to encompass a wider range of chronic conditions such as diabetes, hypertension, and respiratory diseases. Tailoring monitoring solutions to specific patient populations and disease profiles can lead to more effective disease management and improved outcomes.
- 6. User Experience and Human-Centered Design:** Designing IoT health monitoring systems with a focus on usability, accessibility, and user acceptance is essential for ensuring adoption and engagement among patients and healthcare providers. Future work should involve user feedback and iterative design processes to create intuitive and user-friendly interfaces.
- 7. Cost-Effectiveness and Scalability:** Research efforts should be directed towards developing cost-effective IoT solutions that can be deployed at scale across diverse healthcare settings, including resource-constrained environments. This involves optimizing hardware components, streamlining manufacturing processes, and exploring novel business models for sustainable deployment and maintenance.

By addressing these challenges and opportunities, future research in IoT-based health monitoring systems can unlock new possibilities for revolutionizing healthcare delivery, improving patient outcomes, and promoting preventive and personalized medicine on a global scale.

## REFERENCES

- Z. Duan, D. Bai, Y. Zhu, and S. Yang, "IEEE Access " , 8 , 2020, 112797-112808.
- T. Arvind, R. Shriram, S. Keerthana, and K. Anitha ,“ International Journal of Engineering Research & Technology(IJERT)”, 8 , 2019 , 345-349.
- M. Alharbi, M. Alshehri, M. Alanazi, and A. Alamri Journal , “International Journal of Distributed Sensor Networks”,16 , 2020 , 1-9.
- C. Y. Hsu, C. L. Chen, and Y. F. YenJournal , “IEEE Internet of Things Journal”,5,20185965-5973.
- S. Alharbi, H. Almotairi, A. Almoqhim, and M. Khabazi , “Journal of Healthcare Engineering “ , 2019 , 2019 , 1-10.
- I. Gupta , Internet of Things in Healthcare: Technologies, Applications, and Challenges, CRC Press , 2020.
- A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi , Internet of Things for Smart Cities , Springer , 2014.
- "IoT-Based Health Monitoring Systems: A Review" , Medium , [https://medium.com/@AmberJones\\_\\_\\_\\_Official/iot-based-health-monitoring-systems-a-review-85fd00962c3b](https://medium.com/@AmberJones____Official/iot-based-health-monitoring-systems-a-review-85fd00962c3b) , April 25, 2024.
- "The Internet of Things: A survey" , A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash , IEEE Internet of Things Journal , <https://ieeexplore.ieee.org/document/7796698> , 2015.
- "Internet of Things (IoT): A vision, architectural elements, and future directions", J. Gubbi, R. Buyya, S. Marusic, and M. PalaniswamiWebsite: Future Generation Computer Systems , <https://www.sciencedirect.com/science/article/pii/S0167739X12001809> , 2013.
- "Internet of Things: Architectures, Protocols, and Applications" , A. Botta, W. de Donato, V. Persico, and A. Pescapé , IEEE Transactions on Internet of Things , <https://ieeexplore.ieee.org/document/7104239> , 2016.

# APPENDICES

## A.SAMPLE CODE

### i) Import library and pin declaration



```
HeartBeat_v01 (Arduino 1.8.19)
File Edit Sketch Tools Help
HeartBeat_v01
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <PulseSensorPlayground.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);
#define PulseWire A0
#define TempPin A1 // LM35 temperature sensor pin
#define BUZZER_PIN 7
#define LED_PIN 12
#define BASE_THRESHOLD 550
#define MAX_BPM 180
#define MIN_BPM 30
#define AVERAGE_BUFFER 10

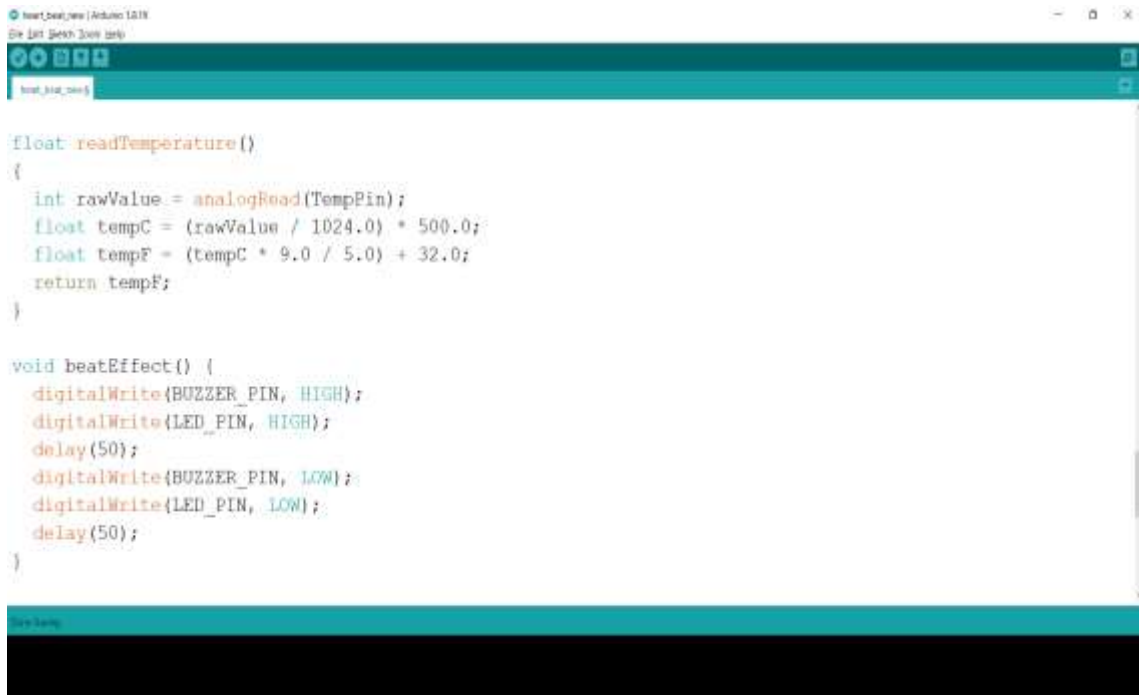
PulseSensorPlayground pulseSensor;
int bpmReadings[AVERAGE_BUFFER];
int readIndex = 0;
float total = 0;
float average = 0;
int BPMcalibration = 0;
```

### ii) Setup function



```
HeartBeat_v01 (Arduino 1.8.19)
File Edit Sketch Tools Help
HeartBeat_v01
void setup()
{
  Wire.begin();
  lcd.init();
  lcd.backlight();
  Serial.begin(9600);
  pulseSensor.unplugInput(PulseWire);
  pulseSensor.setThreshold(BASE_THRESHOLD);
  pulseSensor.begin();
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN, LOW);
  // Initialize the bpmReadings array
  for (int thisReading = 0; thisReading < AVERAGE_BUFFER; thisReading++) {
    bpmReadings[thisReading] = 0;
  }
  lcd.setCursor(0, 0);
  lcd.print("Patient Health");
  lcd.setCursor(0, 1);
  lcd.print("Monitoring System");
  delay(1500);
  lcd.clear();
  calibrateSensor();
}
```

### iii) Buzzer effect and Temperature conversion



```
float readTemperature()
{
  int rawValue = analogRead(TempPin);
  float tempC = (rawValue / 1024.0) * 500.0;
  float tempF = (tempC * 9.0 / 5.0) + 32.0;
  return tempF;
}

void beatEffect() {
  digitalWrite(BUZZER_PIN, HIGH);
  digitalWrite(LED_PIN, HIGH);
  delay(50);
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN, LOW);
  delay(50);
}
```

### iv) Sensor reading



```
void calibrateSensor()
{
  Serial.println("Calibrating sensor. Please wait...");
  delay(1000);
  int sum = 0;
  int validReadings = 0;
  while (validReadings < 100)
  {
    int bpm = pulseSensor.getBeatsPerMinute();
    if (bpm > MIN_BPM && bpm < MAX_BPM) {
      sum += bpm;
      validReadings++;
      delay(10);
    }
  }
  BPMcalibration = sum / 100;
  Serial.print("Calibration completed. BPM Offset: ");
  Serial.println(BPMcalibration);
}
```

## v) Loop function



```
heart_beat_new | Arduino IDE
File Edit Sketch Tools Help

heart_beat_new
void loop()
{
  int myBPM = pulseSensor.getBeatsPerMinute();
  if (pulseSensor.sawStartOfBeat()) {
    if (myBPM > MIN_BPM && myBPM < MAX_BPM) {
      // Averaging the BPM
      total -= bpmReadings[readIndex];
      bpmReadings[readIndex] = myBPM;
      total += bpmReadings[readIndex];
      readIndex = (readIndex + 1) % AVERAGE_BUFFER;
      average = total / AVERAGE_BUFFER;
      beatEffect();
      displayBPMandTemperature(average + BPMcalibration);
    }
  } else {
    digitalWrite(BUZZER_PIN, LOW);
    digitalWrite(LED_PIN, LOW);
  }
}
```

## vi) Printing values

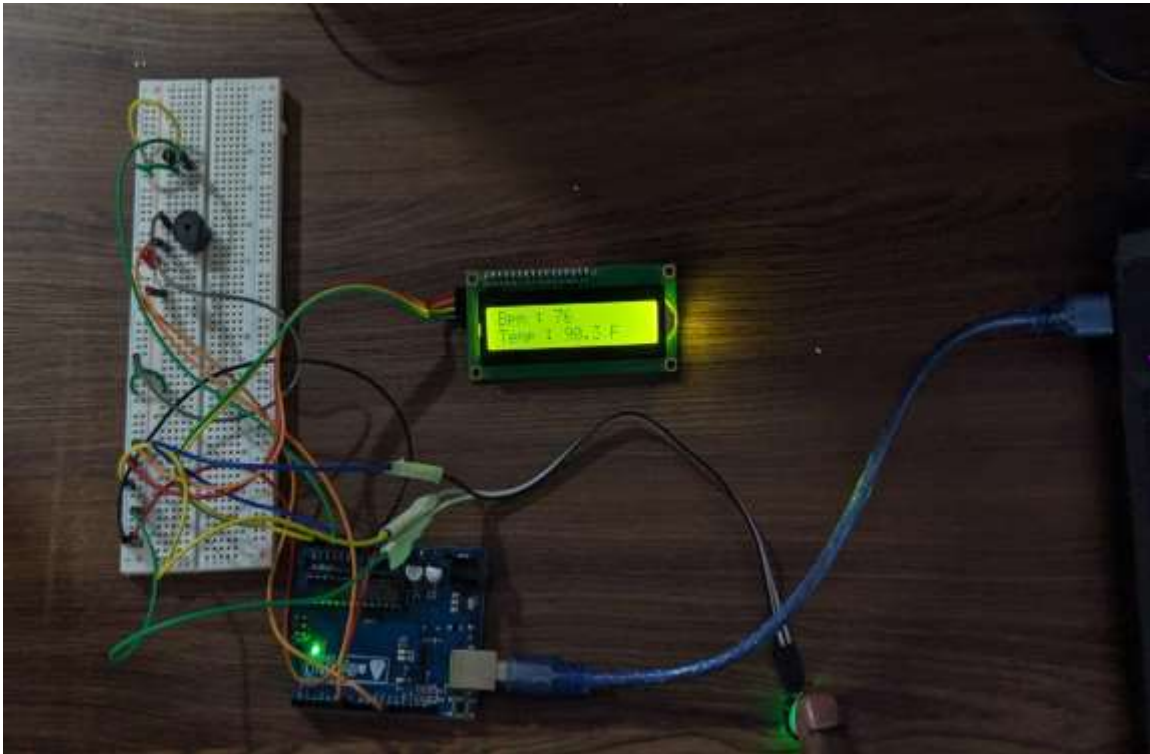


```
heart_beat_new | Arduino IDE
File Edit Sketch Tools Help

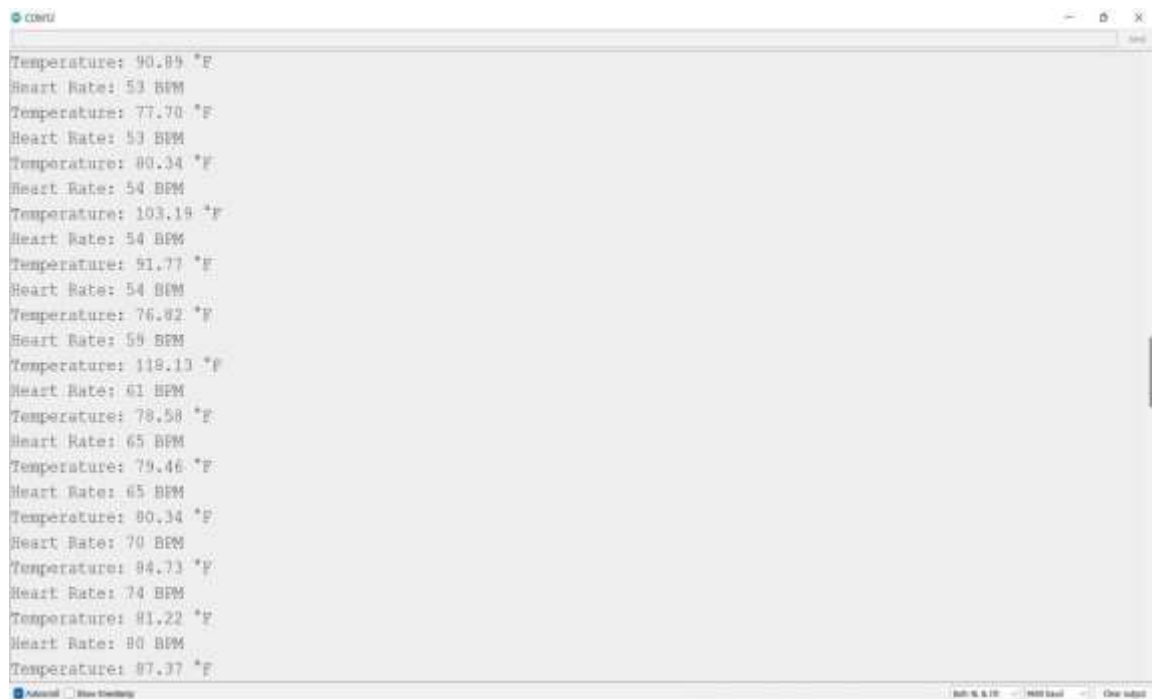
heart_beat_new
void displayBPMandTemperature(float bpm)
{
  lcd.setCursor(0, 0);
  lcd.print("Beats:   ");
  lcd.setCursor(4, 0);
  lcd.print(bpm, 0);
  lcd.print(" BPM");
  float tempF = readTemperature();
  lcd.setCursor(0, 1);
  lcd.print("Temp:   ");
  lcd.setCursor(5, 1);
  lcd.print(tempF, 1);
  lcd.print(" F");
  // Output for Serial Monitor
  Serial.print("Heart Rate: ");
  Serial.print(bpm, 0);
  Serial.print(" BPM, Temperature: ");
  Serial.print(tempF);
  Serial.println(" F");
  // Output for Serial Plotter
  Serial.print(bpm, 0);
  Serial.print(",");
  Serial.println(tempF, 1);
}
```

## B.SCREENSHOTS

### i) LCD output



### ii) Serial Monitor



### iii) Serial Plotter

