Health Monitoring System

Spring 2025

CSCE 430101- Embedded Systems

The American University in Cairo

Ahmed Amin	**900202813**
Mario Ghaly	**900202178**
Youssef Abuzeid	**900202802**

Submitted to:

Dr. Mohamed Sedky

1. Project Overview

1.1 Introduction

This project presents a portable, Arduino-based healthcare monitoring device designed to continuously measure and display a person's heart rate and body temperature. The device is intended to have alert mechanisms for abnormal readings using audio alerts and Bluetooth communication to a paired device. The system targets use cases such as at-home health tracking, low-resource clinics, and wearable devices by enabling real-time vital sign monitoring.

What the project does is detecting heart beats per minute (BPM) through a pulse sensor, measuring body temperature through a MAX30205 temperature sensor, a Liquid-crystal display (LCD) screen for real-time measurements display, a buzzer for alert signaling in case of abnormal readings, and an HC-05 Bluetooth module for wireless interaction with a mobile phone or computer. Users can control the device's behavior remotely via Bluetooth commands, which are the following:

1. Turn on the alert for temperature anomalies only
2. Turn on the alert for heart rate anomalies only
3. Turn on the alert for both temperature and heart rate anomalies
4. Turn off the alert for any anomalies

1.2 High-level Overview

The healthcare monitoring system consists of two main components:

1. **Wearable Component** for handling sensing, processing, and feedback
2. **Application Component** for providing wireless communication with a mobile device

The following figure visualizes the system architecture with its components.

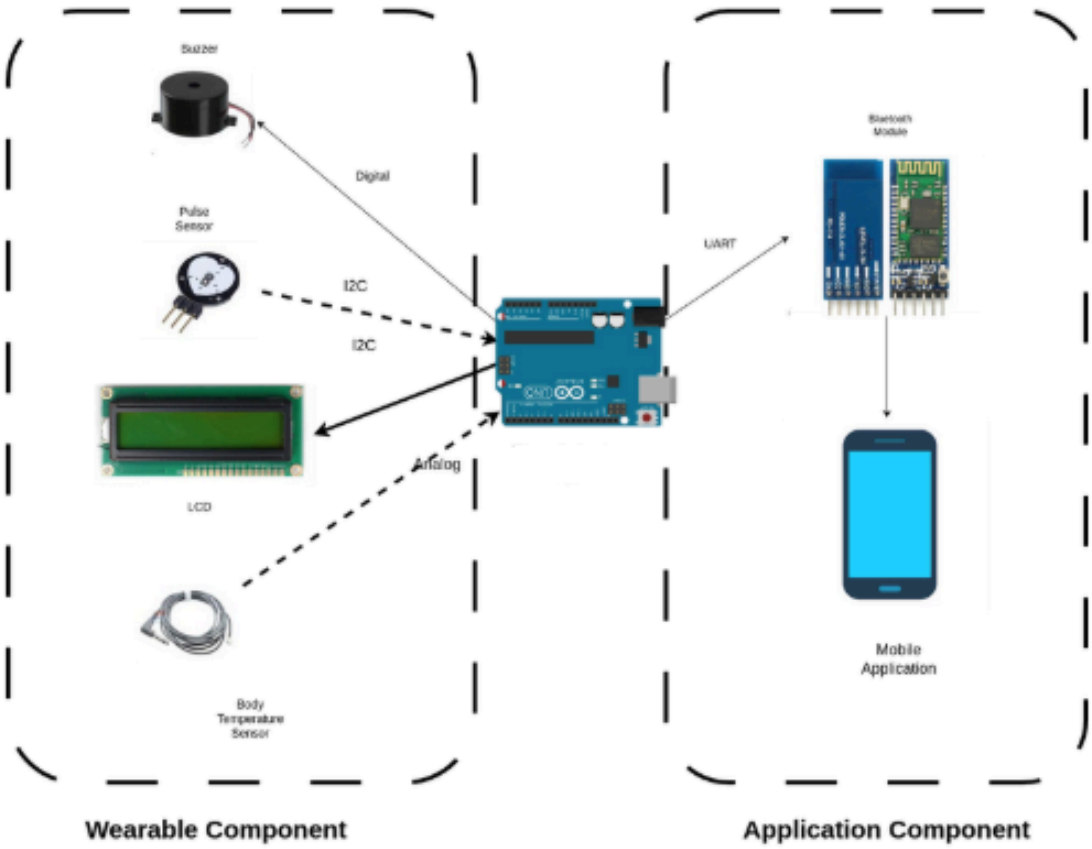

Figure 1: High-level Architecture

Main Project Components:

- **Arduino Uno**

  Arduino Uno is the core embedded board due to its ease of use, availability, and compatibility with various sensors and modules.

- **Pulse Sensor**

  It is connected to an analog input on the Arduino, and it detects heartbeat pulses and transmits raw analog signals that are then converted into a digital value via Arduino's built-in Analog-to-Digital Converter (ADC).

- **Body Temperature Sensor (MAX30205)**

  It communicates via the I2C protocol, and it provides the body temperature data.

- **LCD Display**

  The LCD screen works as an output device to display in real-time both the heart rate and the body temperature using I2C as well.

➔ **Buzzer**

The buzzer is used as a digital output device for triggering alerts when the system is getting an abnormal temperature or pulse rate.

➔ **Bluetooth Module (HC-05)**

It uses UART serial communication to enable wireless communication between the health monitoring system and a mobile application. It acts as both an input and output device, allowing data transmission in both directions for alerts and mode selection.

Additional components to be mentioned are resistors to ensure proper current flow and protect the electronic components within the circuit, and the breadboard for connecting all of those together.

1.3 Implemented Features & Concepts

All the aforementioned described components were successfully implemented as intended from the project proposal. However, the only unimplemented intended feature – in case there was time – is sending the readings and alert mechanisms via wifi or sms instead for better usability. This makes the key embedded concepts covered by this project are the following:

1. **GPIO:** Controls buzzer, LCD, and sensor pins via digital input/output
2. **Serial Communication:** Bluetooth communication via SoftwareSerial
3. **I2C protocol:** Used to interface with the temperature sensor and the LCD
4. **ADC:** Converts analog signals from the pulse sensor to digital values for heart BPM
5. **Interrupts:** For handling Bluetooth input and output and measuring heart BPM
6. **Timers:** To measure time intervals for BPM calculation and periodic Bluetooth data transmission

2. Hardware

2.1 Embedded Board Description

Arduino Uno R3 is the core embedded board used in this project, featuring the ATmega328P microcontroller. This MCU runs at 16 MHz clock speed and provides 2 KB of SRAM and 32 KB of flash memory, suitable for running sensor data acquisition and control algorithms.

The board offers digital and analog I/O pins, including 6 analog inputs that are utilized for sensor interfacing. The Arduino runs a bare-metal firmware programmed through the Arduino IDE, without an operating system, ensuring deterministic real-time control over peripherals.

2.2 Input Device Description

There are three input devices for the system:

1. **Pulse Sensor:**
   An analog sensor connected to one of the Arduino's analog input pins. The sensor outputs a voltage proportional to the detected pulse waveform, which is sampled by the Arduino's ADC. We apply median filtering in software to reduce noise before processing pulse rate.
2. **MAX30205 Body Temperature Sensor:**
   A digital sensor communicating over the I2C protocol with the Arduino acting as I2C master. The sensor's 7-bit I2C address and register map are used to request accurate temperature readings. Communication adheres to the standard I2C start, read, and stop conditions to ensure reliable data transfer, in addition to an extra ACK that is required for this sensor as it was implemented in a low-level manner by reading and writing register values.
3. **HC-05 Bluetooth Module:**
   A module acting as an input device by accepting commands via UART from the paired device. It receives commands from the paired device to choose the alert mode (temperature-only, heart-rate-only, both, or off).

2.3 Output Device Description

There are three output devices for the system:

1. **LCD Display:**

   An LCD connected via I2C bus, allowing efficient two-wire communication. The LCD displays real-time pulse and temperature values to the user.

2. **Buzzer:**

   An active buzzer connected to a digital output pin. It is toggled on or off using digital signals from the Arduino. The buzzer is activated based on alert conditions, abnormal temperature or heart rate, using simple digitalWrite logic.

3. **HC-05 Bluetooth Module:**

   A module communicating over UART serial protocol, enabling wireless transmission of sensors' measurements and alerts to a paired device.

2.4 Supporting Documentation

Based on the components, these are the datasheets needed for the hardware used:

1. Arduino Uno R3 Datasheet
2. MAX30205 Datasheet
3. Pulse Sensor Documentation
4. HC-05 Bluetooth Module Documentation
5. LCD Datasheet

Additionally, below is the full picture for all the circuit connections in hardware.

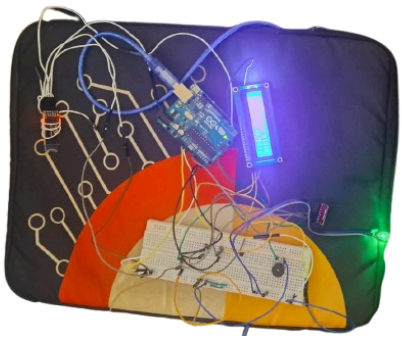

Figure 2: Circuit Connections

2.5 Power Consumption

The system is powered by a 5V regulated power supply sourced from a USB or battery pack. The Arduino Uno and peripherals have moderate power consumption, with the LCD backlight and buzzer as the main dynamic loads.

To optimize energy use in future, the system could minimize buzzer activation duration and give an option of disabling the LCD backlight during idle periods. The Arduino's sleep modes could be implemented in future iterations to further reduce power during inactive phases, making the system more suitable for portable or battery-powered healthcare monitoring applications.

3. Software

3.1 Programming Language

The primary programming language used in this project is C/C++, specifically via the Arduino IDE environment. Arduino's language is essentially C++ with simplified libraries, making it suitable for embedded system development on the Arduino Uno platform. The choice was driven by the following reasons.

1. **Ease of use:** The Arduino IDE and libraries provide a straightforward way to interface with sensors (like the pulse sensor), actuators (like the LCD), and communication modules (Bluetooth HC-05).
2. **Community support:** Extensive online resources, example code, and libraries helped speed up development. The libraries used in this project are the following:
    a. **LiquidCrystal_I2C.h** → to control the LCD via I2C
    b. **PulseSensorPlayground.h** → for pulse sensor data
    c. **SoftwareSerial.h** → to enable serial communication with the Bluetooth module

    For other non-mentioned components – the temperature sensor and the buzzer – low-level implementations were written.
3. **Hardware compatibility:** Arduino's native support for microcontrollers used in our hardware components ensures reliable performance and straightforward debugging.

3.2 Real Timeness

The device operates under real-time constraints due to its continuous monitoring of vital health parameters such as pulse and body temperature, where timely and accurate readings are essential. In this context, "real-time" is defined as a processing window of 5 seconds. This is because each sensor reading is taken every 1 second, and the system computes the median of the last 10 readings to generate a stable and reliable output. The use of median filtering helps mitigate the effects of signal noise, reducing the likelihood of false alerts. This ensures that transient spikes or erroneous sensor values do not trigger unnecessary buzzer activation, which could otherwise lead to inefficiency.

Delays in processing or communication could lead to delayed health alerts, risking patient safety if abnormal readings are not promptly reported. Moreover, it leads to loss of data continuity, causing inaccurate trend monitoring. If the code encounters an unexpectedly large delay, the system may fail to trigger alerts on time or miss critical events. To mitigate this, the code uses non-blocking calls(interrupts) where possible and prioritizes sensor reading and alerting tasks to maintain responsiveness.

3.3 Security

The device's primary security concern lies in its Bluetooth communication, as all other components are hardwired within the wearable unit and are therefore physically secured. Bluetooth introduces potential vulnerabilities if not properly managed. These include:

1. Unauthorized access to health data transmitted over Bluetooth if another nearby device connects without permission.
2. Injection of malicious commands via Bluetooth, such as disabling the buzzer alert, if the attacker successfully connects to the module.

To minimize risks:

1. The Bluetooth module should require pairing with a known device using a PIN before any data exchange.
2. No internet connectivity or external network access is enabled, limiting remote attacks.

Given these measures, the device presents a low risk of exploitation, but users should remain cautious with Bluetooth pairing and environment.

3.4 Original Code

This the link to the GitHub repository with all the necessary code and documentation.
Health Monitoring System GitHub

4. Related Work

4.1 Similar Projects

Below is a non-comprehensive list of similar projects:

1. **IoT-Based Health Monitoring System – Arduino Project Hub**
   Uses an LM35 temperature sensor and a pulse sensor to collect data and upload it to ThingSpeak using an ESP8266 module. Displays results on an LCD and allows remote monitoring via the cloud.
   Link:
   https://projecthub.arduino.cc/rajeshjiet/iot-based-health-monitoring-system-arduino-project-27f2ba
2. **Patient Health Monitoring Using ESP8266 & Arduino – Instructables**
   Tracks temperature and pulse using LM35 and a pulse sensor. Sends data to a web dashboard using ESP8266 and displays locally on an LCD screen.
   Link:
   https://www.instructables.com/Patient-Health-Monitoring-Based-on-IOT-Using-ESP82/
3. **Health Monitoring System Using Arduino UNO R4 WiFi – Hackster.io**
   Measures temperature (MLX90614), heart rate and SpO2 (MAX30102), and ECG signals (AD8232). Sends data to the Blynk app via built-in WiFi for comprehensive monitoring.
   Link:
   https://www.hackster.io/sunnyagarwal/health-monitoring-system-using-arduino-uno-r4-wifi-and-iot-cb9639
4. **Heartbeat and Body Temperature Monitoring Using Arduino – Medium**
   A basic system that monitors heart rate and body temperature using LM35 and pulse sensors, with output to an LCD and the Serial Monitor. No remote communication features included.
   Link:
   https://medium.com/@chawlamahima76/heartbeat-and-body-temperature-monitoring-using-arduino-cf0a339b50f

5. **Smart Heart Rate Monitoring System Using Arduino-Uno – ResearchGate**

   Focuses on portable real-time monitoring using Arduino Uno, LM35, and a pulse sensor. Data is processed and displayed locally, with optional Wi-Fi connectivity.

   Link: https://www.researchgate.net/publication/375593021_Smart_Heart_rate_monitoring_system_using_Arduino-Uno

4.2 Comparison Evaluation

Project	Sensors Used	Connectivity	Data Display	Alerts	Unique Features
Our Project	MAX30205 (Temp), Pulse Sensor	Bluetooth (HC-05)	16x2 LCD	Buzzer (Modes: OFF, TEMP, BPM)	Low-level I2C logic, Median filtering with outlier removal
IoT-Based Health Monitoring System	LM35, Pulse Sensor	Wi-Fi (ESP8266)	LCD, ThingSpeak	None	Cloud dashboard
Patient Health Monitoring (ESP8266)	LM35, Pulse Sensor	Wi-Fi (ESP8266)	LCD, Web app	None	Web-based IoT display
Health Monitoring (UNO R4 WiFi)	MLX90614, MAX30102, AD8232	Built-in Wi-Fi	Blynk App	App Notifications	ECG & SpO2 included
Heartbeat & Temperature Monitoring (Medium)	LM35, Pulse Sensor	None	LCD, Serial Monitor	None	Simple, no connectivity
Smart Heart Rate Monitor (ResearchGate)	LM35, Pulse Sensor	Wi-Fi (optional)	Local Display	None	Real-time portable monitoring

Advantages of our project:

Compared to other health monitoring systems, our project stands out by using a more accurate temperature sensor (MAX30205) and implementing low-level I2C communication without relying on high-level libraries. Unlike many Wi-Fi-based systems focused on cloud integration, our project uses Bluetooth, making it better suited for short-range, low-power applications. It also includes a reliable median filtering approach with outlier exclusion, which many similar systems lack. Additionally, the buzzer system supports three configurable modes—OFF, alert on temperature, or alert on BPM—offering more flexible alerting. While other projects may support more parameters or remote dashboards, our project emphasizes accuracy, configurability, and efficient embedded operation for real-time, local health monitoring.

5. Challenges

1. **Hardware Compatibility:**
   The Pulse Sensor library is specifically designed for the Arduino Uno, which causes issues when using other microcontrollers. The library relies on Timer 2 in the Uno to generate interrupts for reading data from the sensor. However, different boards handle timers differently. For example, using the Arduino Mega may cause the `setup()` function to get stuck in an infinite loop due to timer incompatibility.

2. **Sensor Calibration and Noise:**
   Both the temperature and pulse sensors are highly sensitive and prone to interference and noise, which can lead to inaccurate or outlier readings. To address this, careful calibration and customized signal processing are necessary to ensure reliable and stable data.

6. Future Work

Our future work aims to enhance the system's functionality, reliability, and energy efficiency. Planned improvements include:

- **Energy Optimization:**
  Reduce buzzer activation time and provide an option to disable the LCD backlight during idle periods. Implementing the Arduino's sleep modes will help conserve energy during inactive phases, making the system more suitable for portable or battery-powered healthcare applications.
- **Expanded Sensor Suite:**
  Integrate additional health monitoring sensors such as SpO_2 and ECG for more comprehensive physiological tracking.
- **Signal Processing Enhancements:**
  Improve noise filtering techniques and increase signal accuracy through more advanced processing algorithms.
- **Mobile Application Development:**
  Create a smartphone application to display real-time data and send alerts, improving user accessibility and interaction.
- **Battery Life Optimization:**
  Utilize low-power modes to extend battery life for longer and more efficient operation.
- **Wireless Connectivity Upgrades:**
  Upgrade from basic modules to BLE (Bluetooth Low Energy) or Wi-Fi for faster, more reliable communication.
- **Real-Time Alerts:**
  Implement critical health alerts via SMS or email for timely notification of emergencies.

7. Contributions

Member	Role
Ahmed Amin	● Connected the pulse sensor and tested it carefully. ● Connected the LCD and tested it. ● Wrote the code of the LCD and the pulse sensor. ● Integrate the components and write the processing logic ● Wrote the report with Mario.
Mario Ghaly	● Connected the bluetooth and the buzzer, wrote their code, and tested each of them. ● Did integration testing for all components with Youssif. ● Wrote a partial low level code for the temperature sensor and tested it ● Wrote the report with Ahmed.
Youssef Abuzeid	● Connected the temperature sensor and wrote its low level code. ● Did the calibration for the temperature sensor and tested it. ● Did integration testing of all components with Mario. ● Prepared the presentation and the code documentation.