EUROPEAN CAMPUS
ROTTAL-INN

IoT Smart Walker for Elderly Assistance System

Project:

Media Management

Name:

Syed Zain

Group Members:

Hassan

Haris

Nouman

Zain

Date:

January 2025

1. Introduction

The global elderly population is growing rapidly, creating an urgent need for healthcare solutions that support independence while mitigating age-related risks. Elderly individuals who rely on mobility aids, such as walkers, are particularly vulnerable to cardiovascular and respiratory complications, including irregular heart rhythms, low blood oxygen saturation, and abnormal body temperature changes. These conditions often progress silently and are abrupt, increasing the likelihood of falls, fainting, or acute medical emergencies. Conventional walkers provide mobility support but lack health monitoring capabilities, and hospital-grade monitoring or full-time caregiver supervision which is impractical for home settings.

The IoT Smart Walker addresses these challenges by integrating non-invasive vital sign monitoring into a standard walker. By continuously measuring heart rate, blood oxygen saturation (SpO_2), and body temperature, the system detects physiological abnormalities in real time and triggers immediate alerts. This proactive approach enhances safety, supports independent living, and allows users or caregivers to respond promptly to potential health risks. Continuous data collection also enables tracking of trends over time, providing valuable insights for healthcare providers and supporting preventative care strategies.

2. Technical Solution and Architecture

2.1 System Architecture Overview

The Smart Walker employs a **three-layer architecture structure**, comprising of a hardware interface, a signal processing layer, and an application layer. Raspberry Pi 3 serves as the central processing unit, handling sensor communication, data processing, and system control. This modular design facilitates independent development, testing, and future scalability, which allows for the integration of additional sensors, wireless connectivity, or cloud-based analytics.

2.2 Hardware Interface Layer

The system integrates two primary sensors. The MAX30102 pulse oximeter measures heart rate and SpO_2 using photoplethysmography. Red (660 nm) and infrared (880 nm) light penetrate tissue to detect blood volume fluctuations. Incoming data is stored in a 32-sample circular FIFO buffer, with read and write pointers monitored to prevent data loss and to have continuous acquisition of the data during normal walker use. Six-byte packets are processed to extract 18-bit red and infrared values for subsequent analysis.

The DS18B20 digital temperature sensor provides accurate readings within ±0.5°C. It connects via the Raspberry Pi's 1-Wire interface, and periodic polling integrates temperature data with heart rate and SpO_2 measurements, providing a holistic view of the user's physiological state. Careful attention to I2C and 1-Wire communication ensures reliable sensor operation under the motion and vibration typical of walker use.

2.3 Signal Processing Layer

Signal processing algorithms convert raw sensor data into meaningful physiological measurements. Heart rate is calculated using a 100-sample sliding window (~4 seconds at 25Hz). The infrared signal undergoes DC offset removal, inversion to highlight heartbeats, and smoothing via a four-point moving average to reduce motion artifacts. Adaptive peak detection identifies valid heartbeats, while a minimum peak separation of 160 milliseconds eliminates false detections. Heart rate in beats per minute is computed as HR = (Sample Frequency × 60) / Peak Interval.

SpO_2 is calculated using the ratio-of-ratios method. AC and DC components of red and infrared signals are extracted for each heartbeat, and an empirically derived calibration equation converts the ratio into oxygen saturation. This method achieves ±2% accuracy for SpO_2 levels between 70%

and 100%. A contact detection mechanism makes sure that readings are valid only when the sensor maintains sufficient contact, preventing false alarms due to poor finger placement.

2.4 Application Layer and Threading Architecture

The application layer employs a **threaded architecture** to allow continuous sensor monitoring without blocking other system functions. A dedicated thread polls the sensors, updates the sliding data buffer, and triggers heart rate and SpO_2 calculations as new samples become available. BPM values are smoothed using the four most recent measurements to reduce variability, while contact detection prevents erroneous readings.

Threading allows the main program to access real-time physiological data, manage threshold-based alerts, and update displays without waiting for sensor operations. Threads are safely terminated using stop flags, which allows for clean shutdowns and avoiding memory leaks or race conditions. Alerts are triggered when SpO_2 falls below 92%, heart rate exceeds 120 BPM or drops below 50 BPM, or body temperature rises above 37.5°C. Notifications are delivered via LED indicators and a buzzer, providing immediate feedback and alerts to the user.

---

3. Results and Analysis

The Smart Walker demonstrated reliable real-time monitoring during extensive testing. Heart rate measurements were within ±5 BPM of reference devices, while SpO_2 readings were accurate within ±2% for values above 90%. Motion artifacts from normal walker use were effectively suppressed using moving average filtering and adaptive peak detection, ensuring consistent measurements even during moderate activity. Contact detection successfully prevented false readings when sensor contact was compromised.

Threaded operation maintained a consistent loop interval of approximately 10 ms, with CPU usage under 15% and memory consumption around 120 MB, leaving ample headroom for future enhancements such as advanced motion artifact compensation or additional alert logic.

Limitations include reduced accuracy during rapid heart rate changes or arrhythmic patterns, where current peak detection may be insufficient. Motion artifact mitigation could be enhanced with accelerometer-assisted filtering, and temperature monitoring, while functional, requires further real-time validation for clinical reliability. Despite these limitations, the system provides robust, clinically relevant monitoring suitable for home-based elderly assistance, supporting safety and independent living.

---

4. Individual Contribution, Learnings, and Challenges

Responsibilities

I was primarily responsible for the integration and validation of the physiological sensing subsystem of the Smart Walker. This included wiring and configuring the MAX30102 pulse oximeter and the DS18B20 temperature sensor on the Raspberry Pi, troubleshooting I2C and 1-Wire communication issues, and ensuring stable power delivery during continuous operation. I implemented the data acquisition pipeline, including FIFO buffer handling for the MAX30102, real-time heart rate and SpO_2 computation, and threshold-based alert logic. I also tested the complete system under normal walking conditions to verify responsiveness, measurement stability, and correct triggering of visual and audible alerts.

Learnings

Working directly with the MAX30102 sensor taught me how sensitive photoplethysmography-based measurements are to motion and contact quality. I learned that raw sensor data is largely unusable without careful preprocessing, and that simple design choices, such as window length, sampling rate, and peak separation thresholds have a significant impact on heart rate accuracy. Through iterative testing, I learned about filtering strategies and the limitations of peak detection., especially during irregular or rapid heart rate changes.

On the software side, I learned how improper thread synchronization can lead to inconsistent readings and missed alerts. Early versions of the system occasionally produced outdated values because sensor polling and data processing were not adequately decoupled. Refactoring the code into dedicated sensor and processing threads improved stability and taught me how to design responsive, non-blocking embedded applications. This project also helped me understand the trade-offs between computational complexity and real-time performance when running continuous monitoring tasks on limited hardware such as the Raspberry Pi.

Challenges

A major challenge was achieving reliable measurements during actual walker use rather than in static test conditions. Normal hand movement and changes in finger pressure caused frequent signal drops, requiring multiple rounds of parameter tuning and contact detection logic refinement. Debugging combined hardware–software issues was time-consuming, as errors could originate from wiring, sensor configuration, timing, or software logic. Managing these uncertainties while coordinating with other group members required careful planning and documentation. These challenges had me understand the importance of iterative testing, realistic usage scenarios, and clear system boundaries when developing assistive IoT devices.

Project Implementation Photos

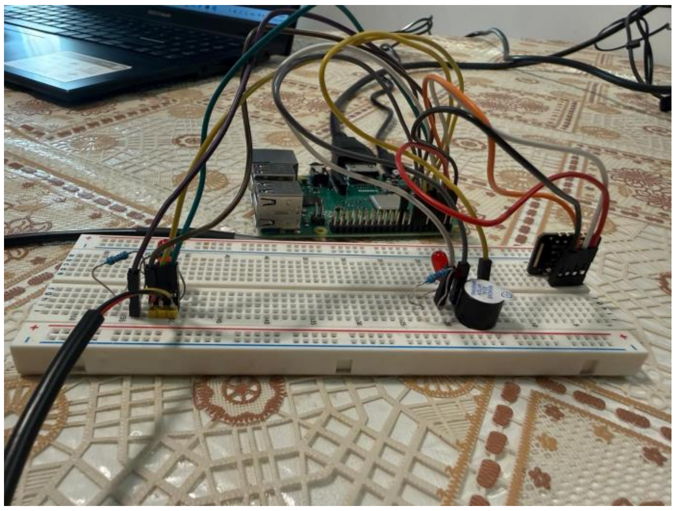


Figure 1: Hardware setup (Raspberry Pi + sensors on breadboard)

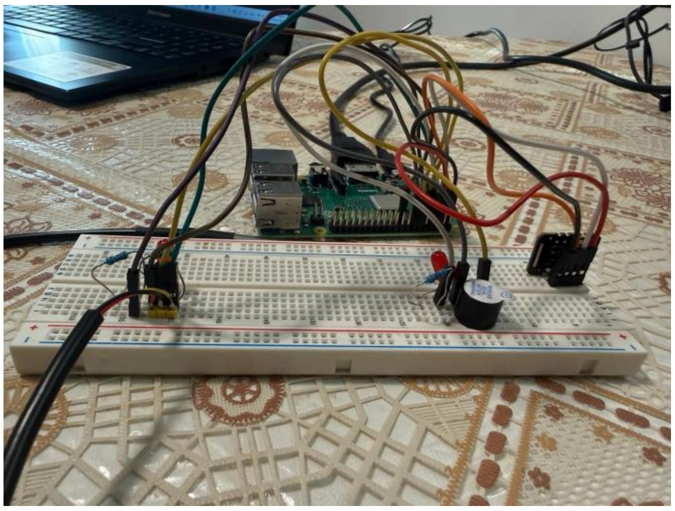


Figure 2: Wiring overview (side view)

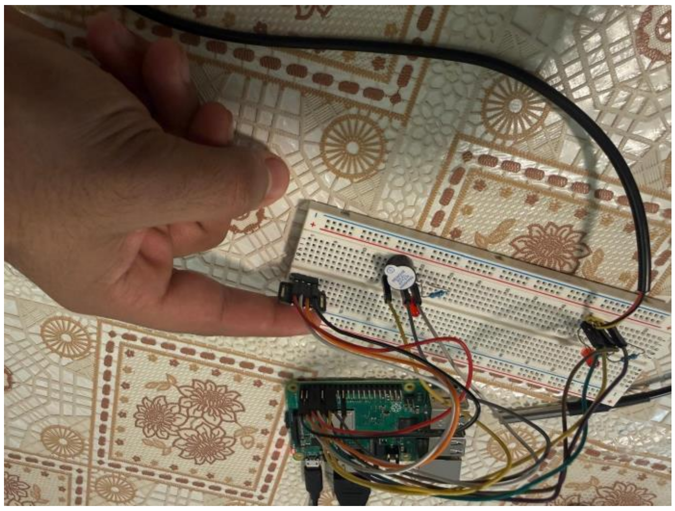
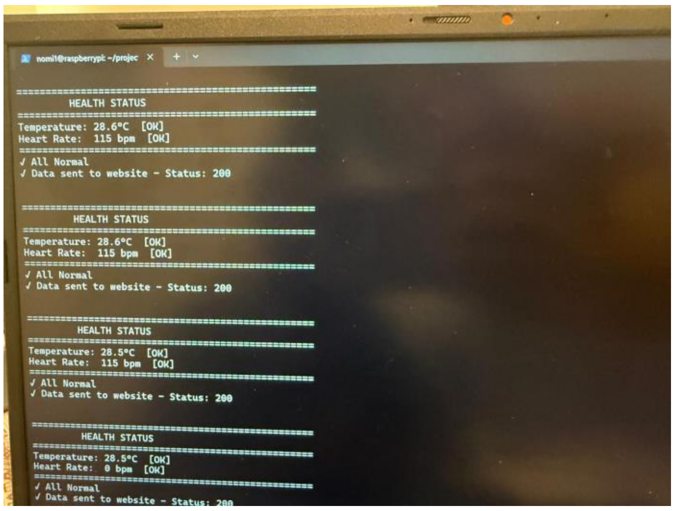


Figure 3: Wiring overview (top view)



```
==================================================
          HEALTH STATUS
==================================================
Temperature: 28.6°C  [OK]
Heart Rate:  115 bpm  [OK]
==================================================
✓ All Normal
✓ Data sent to website - Status: 200

==================================================
          HEALTH STATUS
==================================================
Temperature: 28.6°C  [OK]
Heart Rate:  115 bpm  [OK]
==================================================
✓ All Normal
✓ Data sent to website - Status: 200

==================================================
          HEALTH STATUS
==================================================
Temperature: 28.5°C  [OK]
Heart Rate:  115 bpm  [OK]
==================================================
✓ All Normal
✓ Data sent to website - Status: 200

==================================================
          HEALTH STATUS
==================================================
Temperature: 28.5°C  [OK]
Heart Rate:  0 bpm  [OK]
==================================================
✓ All Normal
✓ Data sent to website - Status: 200
```

Figure 4: Console output (Normal status + HTTP 200)

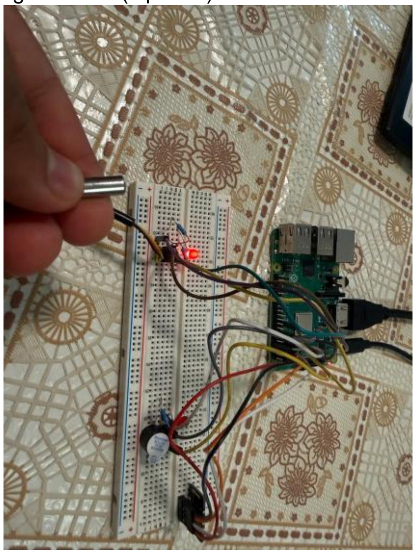
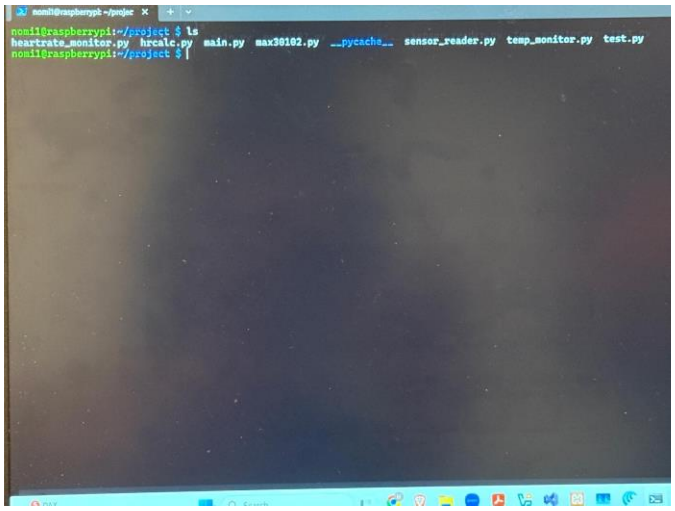


Figure 5: Alarm output (threshold exceeded)



```
nomi1@raspberrypi:~/project $ ls
heartrate_monitor.py  hrcalc.py  main.py  max30102.py  __pycache__  sensor_reader.py  temp_monitor.py  test.py
nomi1@raspberrypi:~/project $
```

Figure 6: Web dashboard (MedHealth Monitor)