IoT-Based Real-Time Health Monitoring System

Rodolfo Daiub
F11315106
NTUST
Asuncion, Paraguay

Iga Wolanin
F11415017
NTUST
Wrocław, Poland

Domonkos Petocz
F11415013
NTUST
Budapest, Hungary

Samadhi Gomez
F11315111
NTUST
Asuncion, Paraguay

***Abstract*—This paper presents a low-cost, IoT-based health monitoring system that integrates Arduino sensors, a Raspberry Pi, and a dashboard. The device measures and displays temperature, humidity, heart rate, and movement on a web dashboard. Testing results demonstrate reliable monitoring and emergency detection, validating this prototype for health monitoring applications.**

***Index Terms*—Internet of Things (IoT), Health Monitoring, Arduino, Photoplethysmography**

I. Introduction

Continuous, real-time health monitoring has applications in patient care, elderly assistance, and early anomaly detection. Traditional monitoring systems often rely on complicated, large, and expensive medical equipment, limited mobility, or manual observation. In contrast, recent advances in IoT technologies enable low-cost and low-power devices that can provide timely health information to caregivers and healthcare systems.

This project presents the design and prototype implementation of an IoT-based health monitoring system that integrates Arduino sensors, a Raspberry Pi gateway, and a web-based visualization interface.

The system collects temperature, humidity, heart rate, and movement data from an Arduino board, then transmits it to the Raspberry Pi. The Raspberry Pi acts as the central IoT node, parsing incoming sensor information, triggering emergency responses, and forwarding data to multiple output modalities. A last key component of this architecture is the Flask-based web dashboard, which provides visualization of patient metrics through an easily accessible interface. This dashboard allows caregivers to remotely monitor vital signs, receive event alerts, and observe emergency requests through a browser-based interface. Because this interface is built on standard web technologies, it is inherently platform-agnostic. Caregivers can access patient data from virtually any device with a web browser.

Initially, the project aimed to develop the device as a smartwatch-style wearable. However, due to physical limitations associated with the available sensors, such as size, power constraints, and rigid wiring, the system could not be miniaturized into a watch form. Because of this, the project pivoted to a monitoring system made for patients with mobility difficulties, focusing on cloud integration and real-time dashboard visualization rather than a wearable form factor. The remainder of this paper is organized as follows:

- Section II reviews related work on IoT health monitoring solutions
- Section III describes the proposed system architecture
- Section IV details the hardware and software implementation
- Section V presents experimental results
- Section VI discusses limitations and future improvements
- Section VII concludes the study

II. Related Work

Wearable technologies, commonly referred to as "Wearable 2.0", have emerged as a promising approach for continuous patient monitoring by integrating sensors, cloud services, and mobile interfaces [1]. These systems demonstrate the potential for high mobility and user comfort, yet, they also face challenges related to sensor size, power consumption, and physical integration—issues that directly influenced the design pivot of the present work.

Several studies propose microcontroller-based IoT health monitoring frameworks in which Arduino or ESP boards capture biometric signals and transmit them to cloud platforms via protocols like MQTT or HTTP. Khan et al. presented an IoT architecture for remote patient monitoring that leverages low-cost sensors and cloud storage to deliver real-time health insights [2]. While these cloud-centric models are effective for global accessibility, they rely heavily on stable network connectivity and often lack mechanisms for local data processing or fast emergency response.

Other research focuses on hybrid IoT architectures where local gateway devices act as intermediaries between sensors and cloud services. Fog and edge computing techniques, as discussed by Mutlag et al., enable gateways such as Raspberry Pi boards to perform local filtering, preprocessing, and event detection before forwarding data to cloud servers [3]. This approach increases system reliability, reduces communication latency, and supports critical real-time applications such as emergency health alerts.

The system developed for this project combines the solutions presented in the literature. The Raspberry Pi gateway handles local event processing, emergency audio signaling, and hosts the web dashboard. This approach was chosen because this approach allows the dashboard to remain operational even without internet access. This was chosen to be local-first, ensuring that data remains accessible even during internet outages. Moreover, the system can be easily enhanced to support viewing the dashboard over the internet, not just on the local network.

III. System Architecture

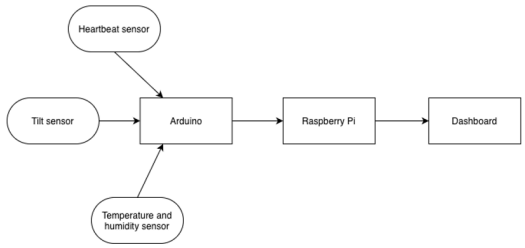

Fig. 1. System Architecture

As mentioned earlier, the proposed system is composed of three major components:

- An Arduino-based sensing unit
- A Raspberry Pi acting as an edge-processing gateway
- A Flask-powered web dashboard for real-time visualization

Fig. 1 illustrates the interaction between these components.

A. Sensing Unit: Arduino Uno board

The sensing subsystem is responsible for acquiring the measured signals. An Arduino Uno is equipped with a temperature and humidity sensor (DHT11/DHT22), a pulse sensor, a tilt sensor, and a button. The tilt sensor's value is read at intervals and aggregated across multiple readings to identify movements. This is bundled into a state that specifies whether the patient is calm or moving around. The button is used as an emergency help button that patients can press to alert caretakers.

The Arduino aggregates sensor readings and formats them into a structured JSON message that includes temperature, humidity, heart rate, movement status, and the button's state. This JSON output is transmitted to the Raspberry Pi via serial communication. This was chosen because the transmitted data is small, and serial is stable and reliable.

B. Gateway: Raspberry Pi

The Raspberry Pi serves as the system's central controller. It continuously listens to the Arduino's serial output, parses incoming JSON messages, performs basic data validation, and initiates system responses based on event conditions.

Critical events, such as a pressed help button or unusual heart rate patterns, trigger local alerts. These can include sound playback through an attached speaker or display images, but we keep it as text for the initial prototype. The Raspberry Pi also updates internal state variables representing the patient's condition and forwards these values to both the web dashboard.

C. Web Dashboard: Flask Application

The Flask-based web application provides real-time visualization of all the metrics measured and calculated by the system. The dashboard displays temperature, humidity, heart rate, motion activity, and emergency status through a clean graphical interface.

The data is delivered to the dashboard via HTTP endpoints periodically updated by the Raspberry Pi. Flask dynamically renders this information, enabling medical staff or caregivers to monitor patient status on any device with any web browser.

D. System Workflow

The complete workflow can be summarized as follows:

- Sensors capture the initial signals
- Arduino packages readings into a JSON message and transmits it over serial
- The Raspberry Pi parses the message, triggers event responses, updates local states, and logs data
- The Flask dashboard retrieves processed data and updates the interface in real time

This modular architecture allows each layer to be extended or replaced independently, making the system suitable for future upgrades such as wearable integration, additional sensors, or implementing machine-learning-based anomaly detection.

IV. Implementation

The implementation of the proposed IoT-based health monitoring system involves two main components: the hardware architecture, which integrates multiple sensors with an Arduino microcontroller, and the software stack running on the Raspberry Pi, which provides data parsing, event handling, cloud synchronization, and real-time dashboard hosting. This section describes the details of both hardware and software components.

A. Hardware implementation

The sensing unit is built using an Arduino Uno connected to a set of sensors and input devices. The two analog sensors are the DHT11 module, which measures environmental temperature and humidity, and the pulse sensor, which provides photoplethysmography (PPG) signals used to estimate heart rate. The digital sensors are the push-button and the tilt sensor. All components are interfaced directly with the Arduino's digital and analog pins, powered by a 5V supply.

B. Software Implementation

a) *Pulse Signal Processing and Heart Rate Smoothing:*
The raw heart rate data obtained from low-cost optical sensors such as the KY-039 or generic pulse sensors is often noisy due to ambient light variation, finger pressure changes, and electrical interference. This proved to be a big challenge in the project. To produce reliable BPM values, the software uses smoothing and peak detection to smooth the sensor signals.

The implementation uses a moving average filter to stabilize the analog readings. Each new pulse sensor value is appended to a rolling buffer, and the average over the most recent N samples is computed. This smoothing step significantly reduces fluctuations and suppresses false peaks.

$$\text{So BPM as B:} \qquad B = \frac{60000}{\Delta_t} \tag{1}$$

To make this more robust, the BPM value is itself smoothed over several beats using an additional averaging stage, reducing jitter and producing a more stable heart rate suitable for real-time monitoring on the dashboard.

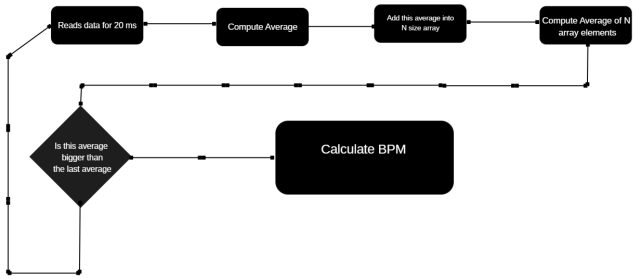

Fig. 2. BPM calculation workflow

b) *Web Dashboard:*
The real-time monitoring dashboard is implemented using Flask, a lightweight Python-based web framework. The dashboard consists of HTML, CSS, and JavaScript components rendered dynamically in response to data updates. A REST endpoint exposes the most recent sensor values, allowing the web interface to fetch updated data at regular intervals using simple AJAX requests.

The dashboard displays temperature, humidity, heart rate, activity status, and emergency alerts using styled UI components.

V. Results

The implemented system was tested to verify whether it can collect and display patient data in real time and provide responsive alerts during emergencies. The results demonstrate correct acquisition of temperature, humidity, heart rate, and movement data, as well as successful transmission to the Flask dashboard.

A. Real-Time Data Visualization

The dashboard successfully displayed smoothed heart rate measurements, environmental readings, and movement detection. The interface updated values at sub-second intervals, a target we wanted to reach for displaying patient data.

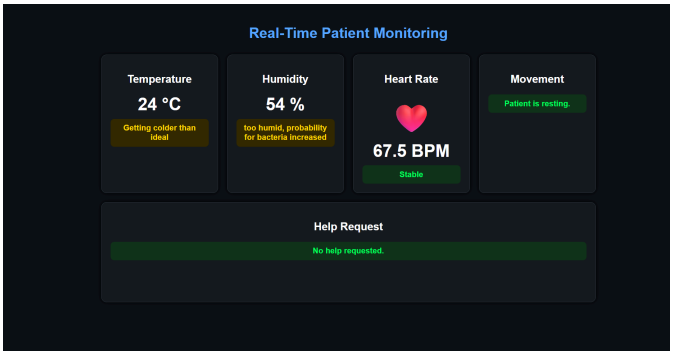

Fig. 3. Real-time monitoring dashboard displaying temperature, humidity, BPM, movement detection, and help-request indicator.

B. Heart Rate Smoothing and Peak Detection Performance

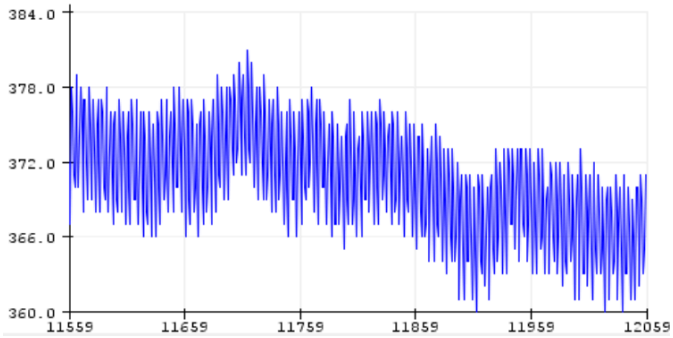

Fig. 4. Sensor without smoothing average

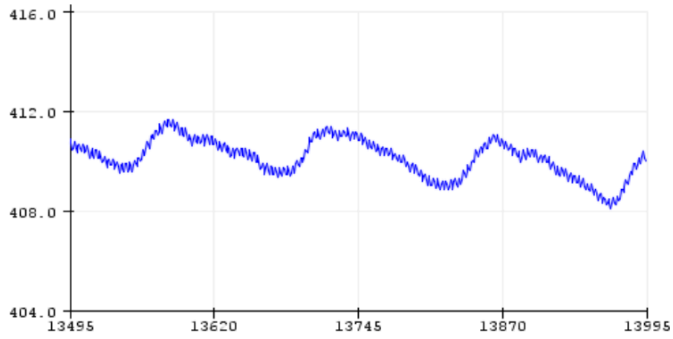

Fig. 5. sensor with smoothing averages

Fig. 4 shows how the pulse sensor produces significant noise. Fig. 5 shows the result of the moving-average smoothing and peak-detection algorithm.

C. Emergency Help-Button Response

When the help button is pressed, the system displays a warning message. This is easily customizable, the action in response to the signal is easily modifiable.

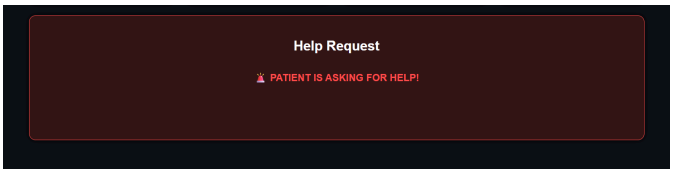

Fig. 6. Help alert triggered

VI. Discussion

The results from the prototype demonstrate that low-cost microcontrollers, combined with lightweight processing and web technologies, can provide a responsive real-time monitoring platform. The system successfully integrates multiple sensors, local processing, and real-time visualization. It offers a complete end-to-end workflow for patient monitoring.

A. Challanges

a) *Blynk Cloud:*

Initially, the platform was going to use Blynk for storing data. However, integration with the Blynk Cloud platform proved to be not ideal for this application. Although Blynk provides a convenient API for remote monitoring, the system frequently experiences issues such as inconsistent virtual pin updates, delayed data synchronization, and difficulties maintaining a stable connection between the Raspberry Pi client and the cloud service. Furthermore, it made the device entirely reliant on an internet connection.

These made real-time monitoring unreliable, especially during periods of fluctuating network performance. Additionally, Blynk's interface imposed structural limitations on how data could be displayed and updated, restricting customization of the user interface and complicating the integration of alert behaviors specific to this project.

Because of these constraints, the system transitioned from relying primarily on Blynk to adopting a fully custom Flask-based web dashboard hosted locally on the Raspberry Pi. This approach offered several advantages. First, it ensured consistent real-time updates independent of internet connectivity, eliminating the latency and reliability issues observed with Blynk. Second, Flask provided full control over UI components, allowing the design of a monitoring interface tailored to the system's needs. These could be custom alert indicators, dynamic color changes, or other expressive layouts.

The shift toward local dashboard development aligns with broader trends in real-time computing, where latency-sensitive applications benefit from running logic close to the data source. In this system, local processing also strengthened emergency responsiveness; when the help button was pressed, both the visual alert and the local audio response were triggered immediately, without reliance on external network services.

b) *Sensors:*

Integrating the heart rate sensor proved to be one of the most challenging aspects of the project. The sensor frequently delivered unreliable or inconsistent data, often producing values that were clearly wrong or wildly fluctuating, with no apparent physiological basis.

Pulse sensor readings exhibited high noise levels due to variations in ambient light and inconsistent finger placement. The moving-average smoothing and peak-detection algorithm mitigated these effects, but the accuracy of BPM estimation remains inherently dependent on sensor quality. For applications requiring medical-grade precision, better hardware would be necessary.

Finally, while the prototype performed well during controlled testing, long-term deployment would require additional features such as persistent data logging, calibration routines, fault detection, and enhanced security measures. Despite these limitations, the system successfully demonstrated the feasibility of the prototype as an IoT health monitoring device.

VII. Conclusions

This work presented the design and implementation of an IoT-based health monitoring system that integrates Arduino-based sensors, a Raspberry Pi for processing, and a custom web dashboard for real-time visualization.

The system successfully captured temperature, humidity, heart rate, and movement data while providing immediate emergency alerts through visual cues. The locally hosted dashboard enabled more stable performance and full customization. Despite constraints on sensor accuracy and other hardware issues, the prototype demonstrates low cost and modularity. It provides a strong foundation for future enhancements, including improved sensors, wearable integration, and advanced data analytics.

VIII. Github repo

https://github.com/rodolfodaiub21/IOT-health-monitoring-system

IX. References

- [1] M. Chen, Y. Ma, Y. Li, D. Wu, Y. Zhang and C. H. Youn, "Wearable 2.0: Enabling Human-Cloud Integration in Next Generation Healthcare Systems," IEEE Communications Magazine, vol. 55, no. 1, pp. 54–61, Jan. 2017.
- [2] A. M. Khan, M. M. Khan, I. U. Rehman, and S. Ullah, "An IoT Framework for Remote Health Monitoring System," in 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), 2020, pp. 1–6.
- [3] A. A. Mutlag, M. K. Abd Ghani, N. Arunkumar, M. A. Mohammed and M. Mohd, "Enabling Technologies for Fog Computing in Healthcare IoT Systems," Future Generation Computer Systems, vol. 90, pp. 62–78, Jan. 2019.