International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1788-1793
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0234

Ai-Driven Heart Rate Variability And Stress Assessment Using Iot System

Deepak S[1], Kedharnath S[2], Thirunavukkarasu M P[3]
[1,2,3,]UG Scholar, Dept. of AI&DS, GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY], [Tiruttani], [Tamilnadu]], India
Emails: *deepakdeepak2284@gmail.com[1], kedharnathai2022@grt.edu.in[2], thirunavukkarasueee2022@grt.edu.in[3]*

Abstract
Heart disease is one of the biggest health challenges of our time, and yet most people only get their heart checked during occasional doctor visits. We built this project to change that. This work describes a wearable heart rate monitor that anyone can put together at a very low cost, using an ESP32 microcontroller and a MAX30102 optical sensor. The device sits on the wrist, reads the pulse continuously using light, and wirelessly sends that data to a small backend system running on a nearby computer. From there, the software figures out not just the heart rate, but also what zone the user is exercising in, whether their stress levels appear elevated, and whether anything looks clinically abnormal. The whole system — from the physical hardware to the Python code that processes the data — was designed to be transparent, modifiable, and genuinely useful. Testing showed solid heart rate readings across the 30 to 220 BPM range, with data reaching the backend in under 120 milliseconds.

Keywords: *MAX30102, PPG Sensor, ESP32, Wearable Health Monitoring, Heart Rate Analysis, IoT, HRV, Real-Time Analytics*

1. Introduction
Cardiovascular diseases account for approximately 18 million fatalities annually, placing them at the top of global mortality statistics according to the World Health Organization. A significant proportion of these deaths are attributable to conditions that could have been identified and managed far earlier, had the right monitoring tools been available to patients in their daily lives. The human heart communicates its distress through measurable signals — elevated resting rates, erratic inter-beat intervals, prolonged post-exercise recovery — yet the absence of continuous monitoring means these early indicators go largely unnoticed by most individuals. Although clinical-grade electrocardiography delivers excellent diagnostic precision, its size, cost, and operational complexity restrict its utility to controlled medical environments. Advances in miniaturised electronics over the past decade have opened the door to continuous physiological monitoring outside clinical walls. Wearable devices can now accompany individuals through their routines, capturing data that was previously accessible only in specialised diagnostic facilities. Among the available non-invasive sensing modalities, photoplethysmography — commonly abbreviated as PPG — stands out for its practicality. The underlying principle relies on directing low-intensity light from a surface-mounted LED into the underlying tissue; the amount of light returned to an adjacent photodetector varies rhythmically with each cardiac cycle because blood volume in the peripheral vessels changes with every beat. This optical signature can be translated reliably into a heart rate measurement without requiring electrode contact or conductive gels. This paper presents a wearable cardiac monitor constructed around two commercially available components: the MAX30102, a dual-wavelength optical sensor small enough to rest against a fingertip, and the ESP32, a WiFi-enabled microcontroller that manages sensor communication and data transmission. The assembled device costs less than USD 15 in components, draws power from a compact rechargeable cell, and continuously forwards heart rate readings to a Python-based analytical backend.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1788-1793
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0234

The system is not a medical-grade diagnostic instrument; rather, it is positioned as an accessible, transparent platform that gives individuals ongoing insight into their cardiac activity that they would otherwise lack entirely. The primary contributions of this paper include: a fully documented, reproducible hardware and firmware design; an analytical backend capable of deriving nine clinically relevant cardiac parameters from raw BPM streams; a low-latency wireless data pipeline with local persistent storage; and empirical validation of the prototype across multiple subjects confirming consistent real-world performance [3].

1.1. Motivation and Scope

Commercial fitness wearables present a fundamental limitation for researchers and technically inclined users: the raw sensor data is inaccessible. The device displays a processed metric, but the signal chain — acquisition, filtering, and storage — operates entirely within a proprietary black box, offering no avenue for customisation or deeper investigation. At the opposite end of the spectrum, research-grade physiological monitoring equipment carries price tags in the hundreds to thousands of dollars, effectively excluding individual researchers, undergraduate students, and under-resourced clinics from meaningful participation. The present work deliberately occupies the gap between those two options. The hardware schematic uses only through-hole and breakout-board components that can be assembled without specialist tools, the embedded firmware is written in the widely familiar Arduino C++ dialect, and the backend processing code is structured, readable Python. The scope of the implementation spans the entire signal chain — hardware assembly, firmware, WiFi transmission, data persistence, alert generation, and session reporting — so that any component can be studied, modified, or replaced independently [2].

2. System Architecture

The system is organized into three layers that each handle a distinct part of the problem. The first layer is the physical hardware — the sensor that touches the skin and the microcontroller that reads it. The second layer is the firmware running inside the ESP32 that turns raw sensor values into meaningful BPM readings and handles the WiFi connection. The third layer is the Python backend running on a computer, which receives the data, analyses it, stores it, and generates alerts when needed. Together, these three layers form a clean pipeline from pulse to insight [1].

2.1.Hardware Architecture

The ESP32 was chosen as the brain of the device because it packs a dual-core 240 MHz processor, built-in WiFi, and Bluetooth into a board the size of a thumb — all for a few dollars. It has more than enough power to handle real-time signal processing, run the WiFi stack, and still leave plenty of headroom for future additions like a display or extra sensors. The MAX30102 sensor talks to the ESP32 over I2C, a simple two-wire protocol that only needs the SDA line on GPIO 21 and the SCL line on GPIO 22. The sensor runs at fast-mode I2C speed, meaning it can push data at 400 kHz without any trouble. Power comes from a 3.7V lithium polymer battery, managed by a TP4056 charging module that automatically handles safe charging over USB and protects the battery from overcharging or overheating. For those who want to know how much battery is left, a simple voltage divider on GPIO 34 gives the ESP32 a readable analog signal that maps to battery percentage shown in Table 1.

Table 1 Hardware Components and Specifications

Component	Specification	Role in System
ESP32 Dev Board	Xtensa LX6, 240 MHz, 4MB Flash, WiFi/BT	Main controller, WiFi gateway
MAX30102	I2C, 100–400 kHz, Red + IR LEDs	PPG signal acquisition
Li-Po Battery	3.7V, 500–1000 mAh	Portable power source
TP4056 Module	1A charge, USB Micro-B	Battery charging & protection
Wrist Strap	Adjustable, 20mm width	Wearable form factor

2.2.Hardware Connection Diagram

Wiring this circuit is genuinely simple. The MAX30102 only needs four wires to the ESP32:

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1788-1793
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0234

power (3.3V), ground, and the two I2C lines. There is an optional fifth connection — the interrupt pin from the sensor to GPIO 19 — which allows the sensor to tap the ESP32 on the shoulder when new data is ready, rather than forcing the firmware to constantly ask. This small addition reduces unnecessary work in the main loop and makes the firmware more efficient. The complete wiring is listed in Table 2.

Table 2. Pin Connection Mapping

Source Pin	ESP32 Pin	GPIO	Function
MAX30102 VIN	3V3	—	3.3V power supply
MAX30102 GND	GND	—	Common ground
MAX30102 SDA	GPIO 21	21	I2C Data line
MAX30102 SCL	GPIO 22	22	I2C Clock line
MAX30102 INT	GPIO 19	19	Data ready interrupt
TP4056 OUT+	VIN	—	5V regulated power
Battery ADC	GPIO 34	34 (ADC1)	Battery % monitor

3. Software Architecture

The software side of this project is split cleanly between the ESP32 firmware and the Python backend. The ESP32 handles everything that needs to happen in real time: reading the sensor, detecting beats, computing BPM, and sending the data over WiFi. The backend handles everything that benefits from a full programming environment: classification, trend analysis, alert logic, and storage. The two sides communicate through a simple HTTP POST request, with the data wrapped in JSON so it is easy to read, extend, and debug [5].

3.1.Embedded Firmware (ESP32 / Arduino)

The firmware is written using the Arduino framework, which gives access to the SparkFun MAX3010x library for sensor control. When the device powers on, it sets up the MAX30102 with a 100 Hz sampling rate, a 411 microsecond LED pulse width, and a 4096 nA ADC range, then connects to the configured WiFi network. These settings were chosen because they give a clean signal with low noise while keeping the LED current at a reasonable level for a battery-powered device. Once everything is initialized, the firmware sits in a loop reading the latest infrared and red values from the sensor's internal FIFO buffer. It runs each IR reading through the check For Beat () function, which watches for the characteristic rise and fall of a pulse waveform. When a beat is detected, the firmware calculates how much time passed since the last one and converts that interval into a BPM value. That value is validated to be within the realistic human range of 30 to 220 BPM and then added to a small four-reading rolling buffer. The average of that buffer is what gets sent to the backend. Every five seconds, if the sensor can confirm a finger is present (which shows up as an IR reading above 50,000), the firmware bundles up the current BPM, the raw sensor values, the battery level, and a timestamp into a JSON message and fires it off to the backend server[4].

3.2.Backend Software Modules

The backend was written in Python using Flask, a lightweight web framework that is well-suited for small data pipelines like this one. Rather than putting everything into a single file, the code is split into six focused modules, each responsible for one part of the job. Table 3 describes what each module does shown in Table 3.

Table 3 Backend Software Module Descriptions

Module	Description
api.py	This is the main entry point for receiving data from the ESP32. It listens for incoming POST requests at /api/heartrate, checks the data for any issues, and then passes it along to the other modules for processing.
processor.py	Once data arrives, this module figures out what the heart rate actually means — whether it falls into bradycardia, normal, or tachycardia range, which exercise zone the user is in, how stressed they might be, and an approximate SpO2 reading.
analytics.py	This module looks at the bigger picture. Instead of just a single reading, it tracks heart rate trends over time, estimates the resting heart rate, and gives a rough idea of the user's HRV and autonomic balance.
alerts.py	If something looks off — like a heart rate that drops below 40 BPM or spikes above 150 BPM — this module raises an alert immediately. It also handles milder warnings for readings that are unusual but not immediately dangerous.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1788-1793
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0234

database.py	All the readings, analysis results, and alerts get saved here using SQLite. Every record is stored with a timestamp so it's easy to look back at past sessions and track how the user's heart health changes over time.
report.py	At the end of a session, this module pulls everything together into a neat, readable report that covers all the key metrics. It's designed to be useful for both everyday users and healthcare professionals.

4. Data Analysis And Metrics

A raw BPM number tells you surprisingly little on its own. Seventy-two beats per minute could mean a healthy resting state, mild exertion, or early stress depending on the context. The analytics module was designed to extract as much meaning as possible from each reading by computing nine separate metrics. Together, these give a far richer picture of what the heart is actually doing [6].

4.1.Heart Rate Classification

The most fundamental step is simply categorizing the heart rate. Clinically, a resting heart rate below 60 BPM is called bradycardia, 60 to 100 BPM is considered normal, and anything above 100 BPM is tachycardia. The system checks every incoming reading against these thresholds and uses the result to decide whether an alert needs to be sent. For most healthy adults at rest, the reading should consistently land in the normal range.

4.2.Heart Rate Zones

For users who are exercising, knowing their heart rate zone is often more useful than the raw number. The system estimates the user's maximum heart rate as 220 minus their age (defaulting to 30 if no age is provided, giving an HR_max of 190 BPM) and then maps the current reading into one of five zones: Resting below 50%, Fat Burn from 50 to 60%, Aerobic from 60 to 70%, Anaerobic from 70 to 85%, and Maximum Effort above 85%. These are the same zones used by fitness coaches and sports scientists worldwide.

4.3.Stress Level Estimation

Psychological stress activates the sympathetic branch of the autonomic nervous system, producing a measurable elevation in heart rate even at rest. Multiple peer-reviewed studies have established this relationship, supporting the use of resting heart rate as a proxy indicator of autonomic arousal and perceived stress load. Drawing on these findings, the system categorises stress into three tiers: Normal for resting BPM at or below 75, Moderate for values between 76 and 90, and High for readings exceeding 90 BPM. While this heuristic is not a validated clinical assessment tool, it offers a practical real-time signal that warrants further attention when elevated persistently [7].

4.4.Approximate HRV

Heart rate variability refers to the natural beat-to-beat fluctuation in the duration of cardiac cycles; it serves as one of the most sensitive non-invasive indicators of autonomic nervous system function. Elevated variability is associated with parasympathetic dominance and overall cardiovascular resilience, whereas reduced variability has been linked to heightened sympathetic activity, acute fatigue, and declining fitness. The proposed system estimates HRV using the SDNN metric — the standard deviation of successive inter-beat intervals — computed over a sliding 60-second observation window. Although this approach does not match the precision of ECG-derived HRV under controlled clinical conditions, it produces a reliable relative index that tracks autonomic shifts across successive monitoring sessions.

4.5.SpO2 Estimation

The MAX30102 acquires simultaneous readings at two distinct wavelengths — red (660 nm) and near-infrared (880 nm) — enabling peripheral oxygen saturation to be estimated through the ratio-of-ratios technique. In the implementation described here, the empirical relation $SpO2 = 110 - 25 \times (AC\_red / AC\_IR)$ is applied to the derived AC components, with the output constrained to the physiologically plausible range of 85 % to 100 %. This expression provides a first-order approximation suitable for trend monitoring in healthy individuals; however, individual optical variability means that accurate absolute readings require calibration against a laboratory-grade pulse oximeter before any clinical interpretation.

5. Results And Discussion

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1788-1793
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0234

5.1.Results

We tested the assembled prototype with three adult volunteers over several sessions. One of the first things we noticed was how quickly the sensor locked on to a signal — in almost every trial, a reliable pulse reading appeared within two seconds of placing a finger on the sensor. Beat detection held steady across the 50 to 130 BPM range, which covers everything from a relaxed resting state through moderate exercise. Above 130 BPM, we occasionally saw missed beats, which we attribute to motion artifact in the rolling average when subjects moved their hands. Table 4 shows the output from a typical 60-second resting session.

Table 4 Sample Analytics Output (Resting, 60-second session)

Metric	Value
Resting Heart Rate (RHR)	72 BPM
Classification	Normal (60–100 BPM)
Heart Rate Zone	Fat Burn (37.9% HR_max)
Stress Level	Normal (≤ 75 BPM threshold)
Cardiac Workload	Normal
Cardiovascular Risk	Low
SpO2 Estimate	97.2% (approx.)
HRV (SDNN approx.)	42 ms (moderate variability)
API Latency	< 120 ms (local WiFi)

On the software side, the numbers were just as encouraging. Data packets sent over the local WiFi network arrived at the backend in under 120 milliseconds consistently, which is well within the five-second window between transmissions. The Flask server processed and saved each incoming record in under 30 milliseconds, and the SQLite database held up across every session without losing or corrupting a single record.

5.2.Discussion

Perhaps the most striking finding is just how much capability fits into such a small budget. The entire hardware cost of this device sits below USD 15, yet it delivers heart rate classification, zone tracking, stress estimation, SpO2 approximation, and HRV insight — features that typically appear only in devices costing ten to thirty times more. The modular design of the backend means individual pieces can be swapped out without touching the rest; for instance, anyone who wants to scale up could replace the SQLite database with PostgreSQL or a cloud time-series service in a single afternoon. That said, motion artifact is a real limitation worth taking seriously. PPG sensors work by detecting tiny changes in how much light reflects back from the skin, so any movement that shifts the sensor or compresses the tissue can produce false readings. Adding an accelerometer and using it to filter out movement-related noise would significantly improve reliability during exercise. The SpO2 formula is also a known simplification — it gives a reasonable estimate for healthy subjects but should be validated against a proper oximeter before being used in any health-critical application. The HRV calculation, while useful for general trend-watching, is also constrained by the hardware. The WiFi stack and other interrupts in the ESP32 introduce small timing jitters that can affect inter-beat interval accuracy. Switching to a hardware interrupt on the MAX30102's INT pin and capturing timestamps with a high-resolution hardware timer would make the HRV estimates significantly more trustworthy.

Conclusion

What started as a question — can we build a genuinely useful heart rate monitor for less than USD 15? — turned into a working, validated system that does considerably more than just count beats. By pairing the MAX30102 sensor with the ESP32 and building a thoughtful Python backend, we ended up with a device that monitors the heart continuously, classifies readings against clinical standards, estimates stress and exercise zones, approximates HRV and oxygen saturation, and flags anything that looks abnormal, all in real time. The nine-metric analytics engine makes the output meaningfully richer than a single BPM reading. The modular architecture means the system can grow — new sensors, more sophisticated machine learning models, cloud storage, multi-user support — without needing to be rebuilt from scratch. And because the entire design is open and well-documented, it is genuinely accessible to students, independent researchers, and clinicians who want a starting point

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 04 April 2026
Page No: 1788-1793
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0234

for their own work.

Looking ahead, the most valuable next steps would be adding an IMU for motion artifact rejection, improving HRV accuracy through interrupt-driven timing, calibrating the SpO2 estimation against a reference device, and exploring anomaly detection models that could catch early warning signs of cardiac events before they become emergencies.

Acknowledgements

The authors gratefully acknowledge GRT Institute of Engineering and Technology, Tiruttani, Tamil Nadu for making their lab facilities available throughout this project. We extend our sincere gratitude to Mrs. Kalavathy D, our project guide, for her invaluable guidance, mentorship, and continuous support throughout the duration of this work.

The work also benefited enormously from the Spark Fun MAX3010x library and the Flask framework, both of which are maintained by generous open-source communities whose contributions made this project far easier to realize.

References

[1]. Maxim Integrated. (2020). MAX30102 High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health. Maxim Integrated Datasheet, 19-7684.

[2]. Espressif Systems. (2023). ESP32 Technical Reference Manual. Espressif Systems Documentation, Version 5.1.

[3]. Castaneda, D., Esparza, A., Ghamari, M., Soltanpur, C., & Nazeran, H. (2018). A review on wearable photoplethysmography sensors and their potential future applications in health care. International Journal of Biosensors & Bioelectronics, 4(4), 195–202.

[4]. Motin, M. A., Karmakar, C. K., & Palaniswami, M. (2019). Selection of empirical mode decomposition techniques for extracting breathing rate from PPG. IEEE Signal Processing Letters, 26(4), 592–596.

[5]. Salahuddin, L., Cho, J., Jeong, M. G., & Kim, D. (2007). Ultra short term analysis of heart rate variability for monitoring mental stress in mobile settings. Proceedings of the 29th Annual International Conference of the IEEE EMBS, 4656–4659.

[6]. Tison, G. H., Sanchez, J. M., Ballinger, B., et al. (2018). Passive detection of atrial fibrillation using a commercially available smartwatch. JAMA Cardiology, 3(5), 409–416.

[7]. Reyes, B. A., Reljin, N., Kong, Y., et al. (2014). Tidal volume and instantaneous respiration rate estimation using a wearable bioimpedance armband. IEEE Journal of Biomedical and Health Informatics, 18(5), 1670–1680.