Journal of COMPUTER AND FORENSIC SCIENCES

CFS 2025, Vol. 4, Issue 1, pp. 20-31
https://doi.org/10.5937/jcfs4-58752
Original research paper

Heart Rate Estimation Using Wearable Sensors and Machine Learning

Nikhil Kumar[1] and Darshan Kumar[2]

[1] Octilyon; nikhil.kumar@octilyon.com

[2] Octilyon; darshan.kumar@octilyon.com

* Corresponding author: nikhil.kumar@octilyon.com

Received: May 9, 2025 • Accepted: June 8, 2025 • Published: June 20, 2025

Abstract: This research explores the development of a heart rate estimation system that integrates wearable sensors with machine learning techniques to achieve high accuracy, low cost, and real-time performance. The project aims to build two-stage phases: a software pipeline for model training and a hardware framework for real-world testing. In the first phase, various machine learning algorithms are trained and fine-tuned using the publicly available PPG DaLiA dataset, which contains physiological data collected during everyday activities. The training process focuses on optimizing performance across different model architectures and configurations. The second phase involves implementing the trained model on a real-time embedded system. An ESP32 microcontroller serves as the central unit to collect data from multiple sensors, including electrocardiography (ECG), photoplethysmography (PPG), galvanic skin response (GSR), temperature, and a 3-axis accelerometer. This data is transmitted wirelessly for preprocessing and inference. The user will see their final predicted heart rate on both an OLED display and a user interface (UI) dashboard.

Keywords: heart rate; PPG; ECG; GSR; temperature; ESP32; machine learning; CNN; LSTM; BiLSTM; GRU.

1. INTRODUCTION

The heartbeat refers to the rhythmic tightening and relaxing of the heart muscles powered by the heart's internal electrical system. This action is essential for pumping blood, which delivers oxygen and nutrients to tissues while removing waste products.

The heart rate (HR) is the number of heartbeats recorded in one minute, typically expressed in beats per minute (BPM). It serves as a critical indicator of how effectively the cardiovascular system is functioning. In healthy adults at rest, the heart rate generally ranges from 60 to 100 BPM, although it can vary based on fitness level, emotional state, and physical exertion. Monitoring heart rate accurately is vital for assessing overall health, detecting medical conditions, and understanding how the body responds to various internal and external factors.

Wearable technology is becoming increasingly essential, especially in areas like fitness and healthcare, where continuous heart rate monitoring plays a critical role. While ECG provides accurate readings, it's impractical for everyday use, leading to the rise of wrist-worn devices using photoplethysmography (PPG), like the Fitbit Charge [2] and Apple Watch

[3]. However, PPG signals are prone to motion artifacts, making heart rate estimation more challenging compared to ECG. Enhancing the reliability of PPG-based predictions remains a key research focus [4].

This paper presents a multi-sensor approach to heart rate estimation by combining data from PPG, GSR, ECG, accelerometer, and temperature sensors. Using machine learning, models are trained on public datasets and validated through a real-time hardware setup powered by an ESP32 microcontroller. The system supports wireless data transmission and real-time analysis, contributing to the development of low-cost, accurate, and scalable solutions for wearable health monitoring.

The system comprises of Heart Rate Monitor sensor AD8232 [6], MAX30102 IR Red LED Photodetector [7], Adafruit-1231 ADXL345 Sensor Image [9], DS18B20 Temperature Sensor Image [10], Tactile Switches [11], and 0.96″ Zoll OLED SSD1306 Display Image [12].

Programming GSR sensors can be learned online [8].

2. SYSTEM DESIGN AND ARCHITECTURE

This system is designed to estimate heart rate by combining wearable sensor data with machine learning models. It is structured around two key phases: the software phase, which focuses on training models using the PPG DaLiA dataset, and the hardware phase, which involves real-time data collection from multiple sensors using an ESP32 microcontroller. Together, these phases enable a practical, real-world solution for accurate and efficient heart rate monitoring [1].

2.1. Software Phase

The software phase of this system focuses on building and validating machine learning models for heart rate estimation using the publicly available PPG DaLiA dataset. To ensure real-world applicability, only data corresponding to sitting and walking activities was selected—two activities feasible for real-time testing in the hardware phase [1].

From the dataset, five sensors were used: ECG, PPG/BVP, accelerometer, EDA, and temperature, chosen based on their relevance to the selected activities. Since the dataset lacked aligned heart rate labels due to different sampling rates, three distinct preprocessing strategies were applied: trimming sensor data to match label count, combining existing and recomputed HR values, and exclusively using HR values recalculated from ECG R-peaks [1].

Neural network models such as CNN, RNN, LSTM, BiLSTM, and GRU were trained using these preprocessed signals to learn patterns linking physiological signals to heart rate. Model performance was evaluated using MAE, MSE, and RMSE, and the best-performing models were saved in .keras format for testing them in the hardware phase. These saved models include the architecture and learned weights but exclude raw training or testing data, preserving only the model's learned behavior [1].

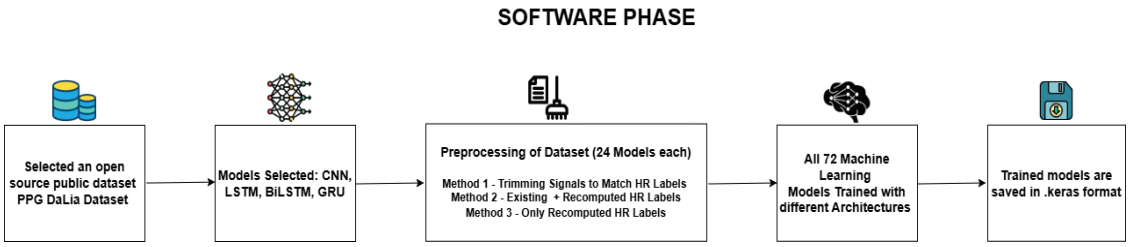

Figure 1. *Software Phase design.*

Three distinct preprocessing methods were applied to align heart rate labels with sensor data: 1. trimming sensor signals to match the existing HR label count, 2. combining existing HR values with those recalculated from ECG R-peaks, and 3. using only recomputed HR labels derived from R-peak intervals. For each method, 24 different models were trained using various combinations of architectures, layers, neuron counts, and hyperparameters, resulting in a total of 72 trained models optimized for heart rate prediction.

The final models were saved in .keras format, preserving the architecture, learned weights, and biases. Notably, the actual training and test datasets were not embedded in the saved files—only the model's learned behaviour was retained. This comprehensive software development phase set the foundation for real-time integration in the hardware phase, enabling scalable, sensor-driven heart rate prediction [1].

2.1.1. Challenges Faced in Software Phase

Several practical challenges were encountered while preparing the PPG DaLiA dataset for training neural networks, particularly around data quality, synchronization, and label alignment. Below are the key issues and how they were resolved:

Challenge 1: Missing Walking Data for Subject S6

Subject S6 had no walking data due to a hardware issue during the original data recording. This was confirmed from the dataset documentation, which stated that only 1.5 hours of data were available for this subject. To prevent runtime errors during training, the code was updated to log a warning and skip missing activity data gracefully [1].

Challenge 2: Mismatch in Sensor Sampling Rates

Each sensor in the dataset had a different sampling rate (e.g., ECG: 700Hz, PPG: 64Hz, EDA & TEMP: 4Hz), leading to a mismatch in data lengths. To resolve this, all five selected sensors (ECG, PPG, ACC, EDA, TEMP) were trimmed to the shortest signal length, ensuring alignment across channels. Only synchronized data related to sitting and walking was retained for training [1].

Challenge 3: Misalignment Between HR Labels and Sensor Data

HR labels were generated independently of sensor streams and had a lower frequency compared to high-frequency sensor data. This caused a significant mismatch between feature and label lengths. To address this, three preprocessing strategies were applied:

Preprocessing Method 1: Trim sensor signals to match HR label count.

Preprocessing Method 2: Combine existing HR labels with those recomputed from ECG R-peaks.

Preprocessing Method 3: Use only recalculated HR labels derived from R-peak intervals.

These approaches ensured proper synchronization between labels and input features, making the dataset suitable for supervised learning [1].

Flow of Software Phase:

Selected Publicly available dataset PPG DaLiA Dataset ➔ Selected CNN, LSTM, BiLSTM, GRU Neural Network Models ➔ Dataset Preprocessing ➔ Model Training ➔ Saved trained models in .keras format.

2.2. Hardware Phase

The hardware phase focuses on building a real-time physiological data acquisition system using a set of wearable sensors interfaced with an ESP32 microcontroller. Acting as the central controller, the ESP32 collects signals from the sensors, performs initial preprocessing, and wirelessly transmits the data for further analysis, thanks to its integrated Wi-Fi capability [1].

The hardware system is composed of the following components:

- ESP32 Microcontroller – core unit for signal collection and wireless communication
- AD8232 ECG Sensor – captures the electrical activity of the heart
- MAX30102 PPG Sensor – used for detecting blood volume changes
- Grove GSR Sensor – measures electrodermal (skin conductance) responses
- ADXL345 Accelerometer – detects movement across three axes
- DS18B20 Temperature Sensor – monitors body temperature
- OLED Display – displays real-time readings or system status
- Tactile Switch – provides manual input control to the system

Together, this setup enables real-time streaming of multi-sensor bio-signals to a connected system for heart rate estimation. The ESP32 acts as the bridge between sensor hardware and software logic, enabling continuous monitoring in a lightweight and portable form.

This section outlines the complete setup of the hardware system, detailing how each sensor is interfaced with the ESP32 microcontroller and how the collected data flows from the sensors to the laptop for further analysis. The architecture is designed to support seamless acquisition of physiological signals in real time, enabling reliable preprocessing and heart rate prediction using trained machine learning models. At the core of the system is the ESP32 microcontroller, which acts as the central hub, connecting and collecting data from five different biosensors. These sensors measure various physiological parameters and transmit their readings to the ESP32. Once received, the data is wirelessly sent to a local computing device, a laptop, for processing and inference. The decision to use a local storage and processing approach was driven by privacy con-

cerns, as data was gathered directly from real human participants. Rather than transmitting sensitive biometric data to the cloud or external servers, all information remained securely on a personal device, reducing risk and having full control over data handling [1].

To ethically manage data collection, all participants were provided with a clear and transparent consent form, outlining what data would be collected, how it would be used, and the steps taken to ensure their privacy. Only after obtaining informed consent was any data recorded [1].

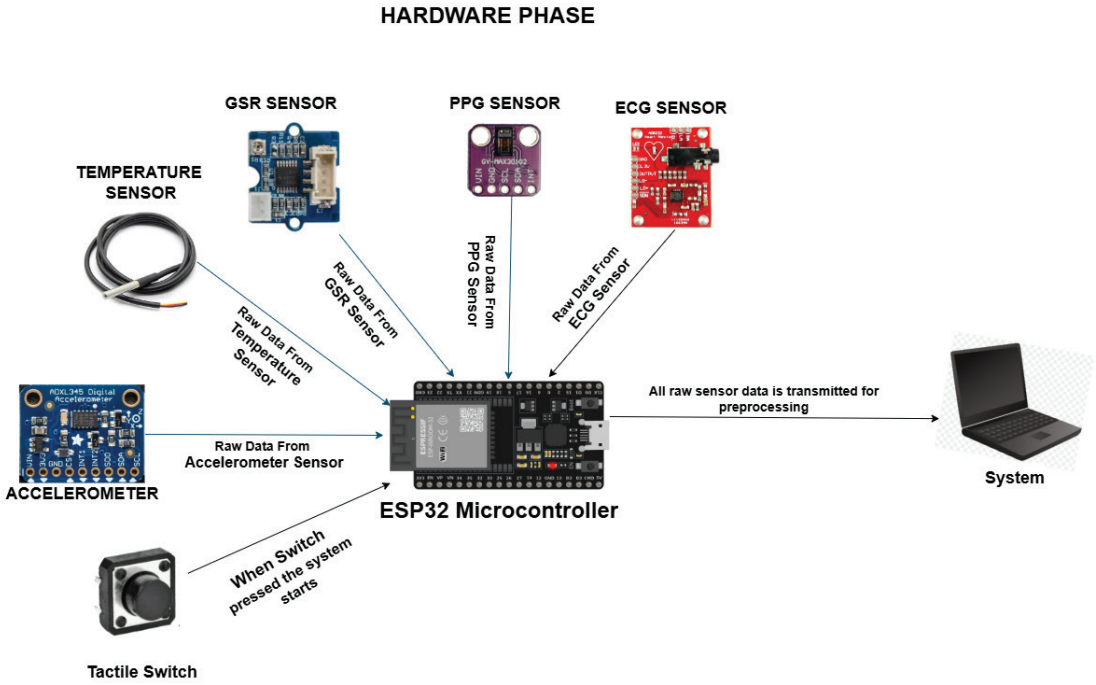

Figure 2. *Architecture Overview of Hardware Phase [5], [6], [7], [8], [9], [10], and [11].*

From the system diagram, it is evident that the ESP32 microcontroller functions as the main unit, interfacing with multiple sensors, an OLED display, and a computer. Once powered on, the ESP32 initializes communication with each connected device, verifying their readiness through a self-check routine. The firmware, developed using the Arduino IDE, is tailored to manage each sensor's specific configuration, such as sampling rate, data acquisition protocol, and power usage. After confirming that all components are properly connected, the system displays the message: "System is Ready, Press Button to Start."

When the tactile switch is pressed, the ESP32 waits for a Wi-Fi connection from the laptop, which acts as the data receiver. Upon successful connection, the microcontroller begins streaming real-time data from the five sensors: MAX30102 (PPG), AD8232 (ECG), DS18B20 (temperature), Grove GSR (EDA), and a 3-axis accelerometer. This raw data is continuously collected over a fixed duration (typically 5–10 minutes) and wirelessly transmitted to the laptop for further processing [1].

2.2.1. Communication Protocols in the Hardware Phase

The sensors used in this hardware setup operate using a mix of communication protocols. While most sensors have built-in analog-to-digital converters (ADCs), two—ECG (AD8232) and GSR (Grove EDA)—output analog signals, which are digitized by the ESP32's internal ADC before being transmitted to a laptop. Regardless of their original format, all sensor data are ultimately transmitted as digital signals, ensuring compatibility with machine learning processing on the receiving system [1].

Three key communication methods were used. I2C (Inter-Integrated Circuit) was employed for digital sensors like the MAX30102, OLED display, and ADXL345. This protocol supports multiple devices using just two lines (SDA and SCL), making it ideal for compact embedded systems. Analog communication was used for ECG and GSR sensors, where raw voltage signals are continuously sampled and converted into digital values. One-Wire communication, used for the DS18B20 temperature sensor, enables data transmission over a single line, simplifying wiring and allowing multiple devices to coexist on the same bus. Each protocol was selected based on the sensor's design and use case, balancing accuracy, power efficiency, and system simplicity [1].

2.2.2. Data Collection Protocol in Hardware Phase

A unique subject ID between 0 and 9 was given to each of the nine subjects, or participants, both male and female, from whom data were gathered during the hardware phase. As was covered in the previous section, the purpose of this data collection was to test and assess the performance of the software phase's pre-trained models using real-time sensor data [1].

As part of the data collection procedure, physiological signals were recorded from subjects performing two specific activities. During the *sitting* activity, subjects remained still in front of the hardware setup while data was continuously collected for 10 minutes. For the *walking* activity, participants walked at a comfortable pace for 15 to 20 minutes before the actual data recording began. To ensure accuracy and reliability, data were collected on multiple occasions for each participant. The overall flow of the hardware phase involved capturing raw data from five different sensors, which was transmitted via the ESP32 microcontroller to a laptop for further processing.

2.3. Integration of Software Phase and Hardware Phase

The integration phase serves as the bridge where the independently developed software and hardware phases are brought together to function as a unified system. In the software phase, a total of 72 machine learning models were trained using the PPG DaLiA dataset, each with varying architectures, layers, neuron counts, and hyperparameters. These models were fine-tuned and stored in .keras format for future inference. On the other hand, the hardware phase involved connecting five physiological sensors to an ESP32 microcontroller, which acted as the central data collection unit. The ESP32 gathered real-time data

from the sensors and wirelessly transmitted it to a laptop. This integration phase is where the real-time sensor data from the hardware phase is fed into the pre-trained models from the software phase, enabling heart rate prediction based on live inputs, thus completing the full cycle from data acquisition to AI-driven inference [1].

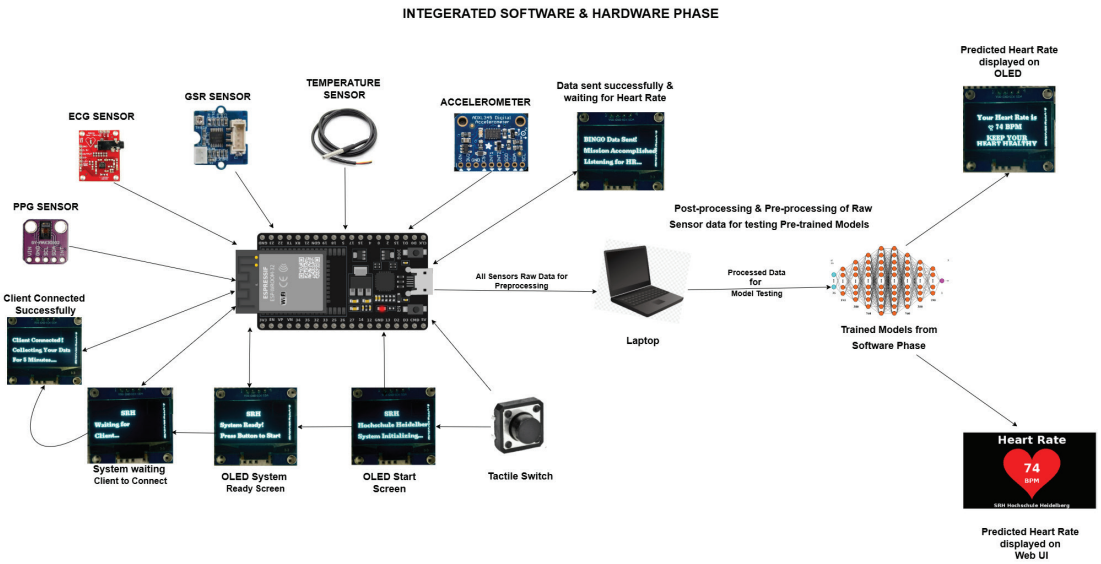

Figure 3. *Architecture of the Heart Rate Estimation System [5], [6], [7], [8], [9], [10], [11], and [12].*

The diagram above presents the complete system architecture of this study as a spectacle of how unseen sensor data collected during the hardware phase is fed into pre-trained .keras models, which were trained on the PPG DaLiA dataset during the software phase, to predict heart rate.

This section outlines the full pipeline for processing real-time physiological signals collected during the hardware phase and preparing them for testing against the pre-trained models developed in the software phase. The dual-phase process includes post-processing the raw sensor data for consistency and noise removal, followed by pre-processing to align with model input requirements [1].

Once data is transmitted from the ESP32 microcontroller to the laptop via Wi-Fi, a custom Python script stores it in .csv and .pkl formats. This raw data contains sensor signals from ECG, BVP, EDA, temperature, and accelerometer sensors, each with distinct sampling rates and potential noise due to motion or environmental factors.

1) Storing Incoming Sensor Data: Incoming raw data is saved as raw_sensor_data.csv, containing timestamps and unfiltered sensor values. This storage preserves the original structure for processing.
2) Signal Cleaning & Preprocessing: To enhance signal quality, various filtering techniques are applied:
   - Bandpass Filtering: ECG (0.5–50 Hz) and BVP (0.5–15 Hz) are filtered to retain heart-related frequencies. EDA and temperature signals are smoothed to remove spikes.
   - Notch Filtering: A 50 Hz notch filter is used to remove the powerline noise.

- Z-Score Normalization: All signals are standardized to mean = 0 and std = 1 for balanced model input. The cleaned data is saved as processed_sensor_data.csv.

1) HR Label Generation via Sliding Window: Heart Rate (HR) labels are computed using an 8-second sliding window (with 2-second shifts) by detecting peaks from ECG (R-peaks) or BVP (as a fallback). R-R intervals are calculated, and the formula HR = 60 / interval (sec) is used to generate BPM values. Each HR label represents the average HR within an 8-second window, ensuring smooth transitions.
2) Synchronization & PKL File Creation: Since sensors operate at varying sampling rates (e.g., ECG at 700 Hz, BVP at 64 Hz, EDA/TEMP at 4 Hz), data is synchronized using timestamps. All sensor readings within an 8-second window are assigned the same HR label. This prevents inconsistencies and enables aligned model input [1].

Each .pkl file contains:

- Time-aligned ECG, BVP, EDA, TEMP, ACC signals
- HR labels matched to the corresponding 8-second windows
- This design manages sensor-sensor and sensor-label mismatches effectively and ensures compatibility with model expectations.

1) Model Prediction using Pre-trained Models: The processed .pkl files are organized subject-wise, each containing multi-session recordings. These are loaded and passed to the 72 pre-trained models (CNN, RNN, LSTM, BiLSTM, GRU) saved in .keras format from the software phase. The models take the sensor data and predict HR values in real time.
2) Real-Time Visualization: Predicted HR values are displayed both on the OLED screen connected to the ESP32 and on a Web UI dashboard [1].

Final Flow of the entire system:

Five Sensors raw data ➔ ESP32 ➔ Laptop (Data Preprocessing and .pkl creation) ➔ Loading PKL file to Pre-trained Models of Software Phase ➔ HR Prediction ➔ Results Displayed on Web App UI and OLED.

3. RESULTS

This section highlights the results obtained from the heart rate estimation system, covering both the software and hardware phases. During the software phase, model performance was assessed using standard evaluation metrics—Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE)—to measure the accuracy of predictions on the training dataset. In the hardware phase, real-time sensor data collected from participants was used to test these pre-trained models, and the same metrics were applied to evaluate how well each model performed under real-world conditions [1].

3.1. Results of the Software Phase

A total of 72 models were evaluated across three different preprocessing strategies, with 24 trained models per method. The most accurate and reliable outcomes from these trials, based on key performance metrics, are highlighted below.

CNN Best Performing Trial of Preprocessing Method 2:

Architecture – There is 1 Input Layer; 4 CNN layers with 256, 128, 64, and 32 neurons with ReLU activation function, kernel size 2, 1 Maxpooling Layer, 1 Dense Layer with 100 numbers of neurons with ReLU activation function, 1 Flatten Layer, and 1 Output Layer.

MAE – 10.655859042778024, MSE – 242.41237147029514, RMSE – 15.569597665652608.

LSTM Best Performing Trial of Preprocessing Method 1:

Architecture – There is 1 Input Layer; 3 LSTM Layers with 128, 64, and 32 neurons with a dropout of 0.2, 2 Dense layers with 128 and 64 numbers of neurons with ReLU activation functions and a dropout of 0.2, and 1 Output Layer.

MAE – 8.07427715704232, MSE – 156.9656917719686, RMSE – 12.528594964000098.

Bi-LSTM Best Performing Trial of Preprocessing Method 2:

Architecture – There is 1 Input Layer, 2 BiLSTM layers with 8,4 neurons with a dropout of 0.2, 1 Dense Layer with 16 numbers of neurons with ReLU activation functions with a dropout of 0.2, and 1 Output Layer.

MAE – 7.3934866673903405, MSE – 137.09411541188987, RMSE – 11.708719631620268.

GRU Best Performing Trial of Preprocessing Method 2:

Architecture – There is 1 Input Layer, 4 GRU layers with 256, 128, and 64 neurons with a dropout of 0.5 and 0.3, 2 Dense layers with 128 and 64 numbers of neurons with ReLU activation functions, and 1 Output Layer.

MAE – 8.893359010141099, MSE – 181.59074108877368, RMSE – 13.475560882158995.

Bi-LSTM Best Performing Trial of Preprocessing Method 3:

Architecture – There is 1 Input Layer, 2 BiLSTM layers with 64,32 neurons with a dropout of 0.2, 1 Dense Layer with 50 numbers of neurons with ReLU activation functions, and 1 Output Layer.

MAE – 1.9532807834843728, MSE – 22.381587964758886, RMSE – 4.730918300368216.

3.2. Results of the Hardware Phase

Real-time data from all five sensors was collected across multiple sessions involving nine individual subjects. This data was tested against all 72 trained models—spanning three different preprocessing techniques—with the best-performing results are mentioned below.

CNN Best Performing Trial of Preprocessing Method 2:

Architecture – There is 1 Input Layer; 4 CNN layers with 256, 128, 64, and 32 neurons with ReLU activation function, kernel size 2, 1 Maxpooling Layer, 1 Dense Layer with 100 numbers of neurons with ReLU activation function, 1 Flatten Layer, and 1 Output Layer.

MAE – 41.149836527493214, MSE – 1886.438650208886, RMSE – 43.43315151136152.

LSTM Best Performing Trial of Preprocessing Method 1:

Architecture – There is 1 Input Layer; 3 LSTM Layers with 128, 64, and 32 neurons with a dropout of 0.2, 2 Dense layers with 128 and 64 numbers of neurons with ReLU activation functions and a dropout of 0.2, and 1 Output Layer.

MAE – 24.73506383666422, MSE – 733.1327101455512, RMSE – 27.076423510972624.

Bi-LSTM Best Performing Trial of Preprocessing Method 3:

Architecture – There is 1 Input Layer, 2 BiLSTM layers with 64 and 32 neurons with a dropout of 0.2, 1 Dense Layer with 50 numbers of neurons with ReLU activation functions, and 1 Output Layer.

MAE – 25.93604230729582, MSE – 813.2612413361254, RMSE – 28.517735557651232.

4. CONCLUSION

This thorough study of heart rate estimation using wearable sensors and machine learning models integrates hardware and software elements for physiological monitoring in real time. The research successfully developed a dependable system that can collect physiological data, preprocess data, train deep learning models, and conduct real-time system testing. This research aimed to bridge the gap between collecting unprocessed sensor data and reliably estimating heart rate for practical applications by leveraging embedded technologies and machine learning [1].

In the software phase, deep learning models such as CNN, LSTM, BiLSTM, and GRU were trained. A total of 72 distinct model trials were carried out to assess different architectures and hyperparameter setups. The models that performed best were selected using their Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE).

Following a successful software implementation, the hardware phase was used to evaluate real-time testing. Real-time physiological data was wirelessly sent to a PC for further

processing using an ESP32 microcontroller that was interfaced with the worn sensors. The trained models were then applied to the sensor data in order to estimate heart rate.

In summary, this work combines machine learning and embedded technologies to offer a strong basis for AI-driven health monitoring systems. The effective deployment of this system demonstrates its potential to enhance wearable healthcare devices by guaranteeing consistent, dependable, and real-time heart rate monitoring in real-world scenarios [1].

5. FUTURE SCOPE

In the future, this system can be extended to support a broader range of human activities beyond just sitting and walking. One major improvement would be miniaturizing the hardware into a compact, wrist-worn form factor, making it more practical and user-friendly for daily wear. Furthermore, by incorporating extra sensors to capture additional health indicators such as respiration rate, SpO2, and other vital signs, the system can evolve into a complete health monitoring solution.

Expanding the range of physiological inputs not only improves model accuracy but also opens doors to predicting a wider set of health-related metrics, such as cardiovascular health, oxygen saturation, and sleep quality, through advanced AI models. This creates a more intelligent and responsive system for supporting overall well-being.

In addition, the data collected from the wearable device could be securely uploaded to cloud storage in an encrypted format. Each user would have personalized access to their own health dashboard via a secure login, ensuring that the data remains private while still being readily available to the individual when needed.

FUNDING:
This research received no external funding.

ACKNOWLEDGMENTS:
We would like to thank all the participants for participating in this study's data collection process.

INSTITUTIONAL REVIEW BOARD STATEMENT:
Not applicable.

INFORMED CONSENT STATEMENT:
Informed consent was obtained from all subjects involved in the study.

CONFLICTS OF INTEREST:
The authors declare no conflict of interest.

REFERENCES

[1] Nikhil Kumar, Heart Rate Estimation Using Wearable Sensors and Machine Learning, Master's thesis, Dept. of Information and Technology, SRH Hochschule Heidelberg, Heidelberg, Germany, 2025.

[2] Fitbit. Fitbit Charge 3, 2018. Available online: https://www.fitbit.com/charge3

[3] Apple. Apple Watch Series Official Website. 2019. Available online: https://www.apple.com/lae/watch/

[4] A. Reiss, I. Indlekofer, P. Schmidt, and R. Stiefelhagen, "Deep PPG: Large-Scale

Heart Rate Estimation with Convolutional Neural Networks," Sensors, vol. 19, no. 14, p. 3079, Jul. 2019. Available: https://doi.org/10.3390/s19143079

[5] DOIT ESP32 Devkit V1 Wi-Fi Development Board – Top Straight View, CircuitState, Sep. 2023. [Online]. Available: https://www.circuitstate.com/wp-content/uploads/2023/09/DOIT-ESP32-Devkit-V1-WiFi-Development-Board-Top-Straight-View-CIRCUITSTATE-Electronics-1.jpg

[6] SparkFun Electronics, Heart Rate Monitor sensor AD8232, SparkFun, Available: https://cdn.sparkfun.com/assets/learn_tutorials/2/5/0/HeartRateBoardFront.jpg

[7] MAX30102 IR Red LED Photodetector, How2Electronics. [Online]. Available: https://how2electronics.com/wp-content/uploads/2024/02/MAX30102-IR-Red-LED-photodetector.jpg

[8] All you need to know about GSR sensor – Galvanic Skin Response guides and projects, Seeed Studio[Online]. Available: https://www.seeedstudio.com/blog/2022/03/07/all-you-need-to-know-about-gsr-sensor-galvanic-skin-response-guides-and-projects/

[9] Adafruit-1231 ADXL345 Sensor Image, Distrelec, 2025. [Online]. Available: https://media.distrelec.com/Web/WebShopImages/landscape_large/9-/01/Adafruit-1231-30139029-01.jpg

[10] DS18B20 Temperature Sensor Image, Elektro Turanis, 2025. [Online]. Available: https://elektro.turanis.de/html/prj054/ds18b20.jpg

[11] "Tactile Switches 101," SameSky Devices, 2025. [Online]. Available: https://www.sameskydevices.com/blog/tactile-switches-101

[12] "0.96 Zoll OLED SSD1306 Display Image," Shopify, 2025. [Online]. Available: https://cdn.shopify.com/s/files/1/1509/1638/products/096-zoll-oled-ssd1306-display-i2c-128-x-64-pixel-kompatibel-mit-arduino-und-raspberry-pi-562312.jpg?v=1679397959

Dataset Reference:

"PPG DaLiA Dataset," Kaggle, 2025. [Online]. Available: https://www.kaggle.com/datasets/ameersifat53/ppg-dalia-dataset