Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Enhanced Ai-Driven Raspberry Pi–Based -Embedded System For Health Monitoring

Prajkta Dhananjay Dangat [1], Dr. Archana Rajesh Date[2], Dr. Sudhir Narsing Divekar[3], Satish Ashok Shelke[4]

[1]Department of E&TC, HSBPVT's GOI Faculty of Engineering, Kashti, Ahilyanagar, India
[2]Associate Professor, Department of E&TC, HSBPVT's GOI Faculty of Engineering, Kashti, Ahilyanagar, India
[3]Head of Department, Department of E&TC, HSBPVT's GOI Faculty of Engineering, Kashti, Ahilyanagar, India
[4]Assistant Professor, Department of E&TC, HSBPVT's GOI Faculty of Engineering, Kashti, Ahilyanagar, India
Mail Id- prajktadangat@gmail.com[1], archanadate@gmail.com[2], sndivekar.pp@gmail.com[3], satishshelke7875@gmail.com[4]

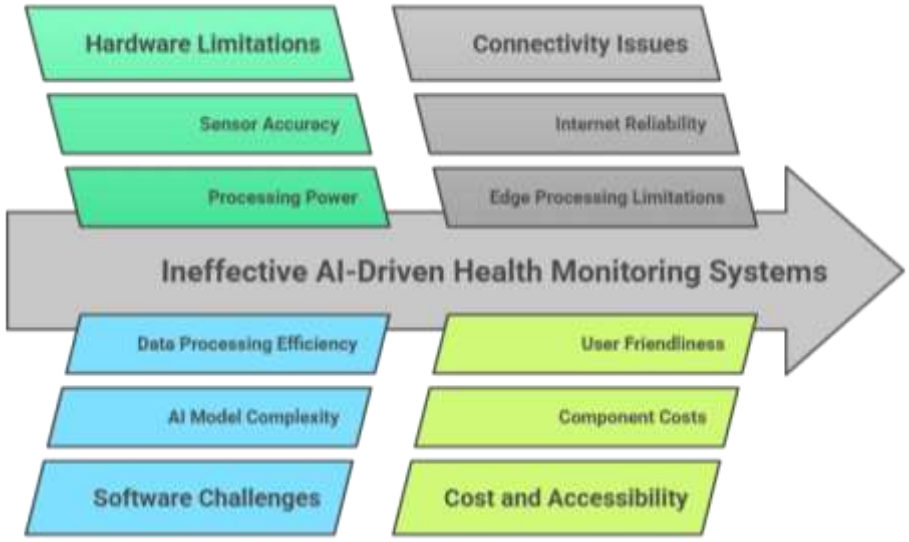
Abstract
The Enhanced AI-Driven Raspberry Pi-Based Embedded Health Monitoring System is expected to solve the increasing demand of continuous health monitoring, particularly to individuals with chronic illnesses or limited access to healthcare services. The proposed project is an inexpensive yet effective health monitoring system based on Raspberry Pi and AI to measure such vital health parameters as heart rate, temperature, and SpO2. The system combines the sensors and real-time data processing with Raspberry Pi, which does not require constant cloud reliance. The AI methods categorize the health conditions as normal, warning, and critical and notify the user in case of abnormal readings. The low cost of the system, combined with the popularity of sensors and the flexibility of Raspberry Pi, makes it practical to use in education and small-scale healthcare. This study highlights the importance of the incorporation of AI into embedded health monitoring systems, its potential, and its limitations. This project will start with conceptual planning and then hardware and software integration to ensure it develops a working prototype that can be used to improve real-time health monitoring and early disease detection.

Keywords: Health Monitoring, Raspberry Pi, AI-Driven System, SpO2, Heart Rate, Temperature, Embedded System, Real-Time Data

1. Introduction

The past years have witnessed a transformation in healthcare monitoring. Previously, the majority of health measurements could be performed only at the moment of a visit to the hospital or clinical checkups [1]. The patients were required to visit physicians or health care facilities to record vital body measurements like heart rate, body temperature, oxygen level and blood pressure [2]. Nevertheless, this technique does not always apply to the older population, the chronic illness patients, or individuals residing in remote locations. Regular observation is required by many patients, and it may be expensive, time-consuming, and challenging to visit the hospital frequently [3]. As such, real-time and intelligent health monitoring has become a significant requirement in contemporary healthcare. The theme of the article Enhanced AI-Driven Raspberry Pi-Based Embedded System to Health Monitoring is the creation of an effective, low-cost, and smart system capable of tracking health status with the help of sensors, embedded hardware, and artificial intelligence methods [4]. The goal of such a system is not to substitute doctors, but to aid in early warning, timely response, and improved patient care. The system can be used to detect any abnormal changes in the condition of a patient before they escalate by gathering health information on a regular basis [5]. In the first stage of this project, a thorough examination of the literature was conducted. The wearable health devices, embedded monitoring system, edge computing, artificial intelligence in healthcare, and low-power hardware design were all a part of the literature review [6]. This review indicated that researchers have proposed a number of health monitoring systems. There are wearable sensor-based systems and mobile app-based systems or cloud-based systems [7]. Nevertheless, most of these systems are still in theory or in concepts. They explain the concept of smart monitoring, but fail to illustrate the entire process of simulation to actual hardware implementation [8].

Fig. 1 Challenges in Developing AI- Driven Health Monitoring Systems



This was the primary impetus of the project. A project at the student level must not just talk about the concept, but it must also demonstrate how the system can be designed, developed, tested and improved. Thus, the current project is dedicated to the creation of a working prototype that will be a combination of hardware and software [9]. The system is supposed to gather real time health information, analyze it and give useful information regarding the health of the user [10]. The primary controller of this embedded system was chosen as Raspberry Pi. It is an effective and dynamic prototyping platform [11]. It has support of various programming languages, sensors and communication modules. Raspberry Pi is more powerful in processing than a simple microcontroller and it can execute more sophisticated software [12]. This renders it to be applicable in AI-based health monitoring applications. It is also easy to use, good in balance to cost and performance. The system can also be linked to various biomedical sensors that will gather health related information. Such sensors can be a heart rate sensor, temperature sensor, pulse oximeter sensor and others [13]. The values obtained are forwarded to Raspberry Pi to process. The system is meant to process data on the local server rather than transmitting all the data to a cloud server. This is referred to as edge processing. It minimizes the need to be connected to the internet and enables quicker response [14]. This system is significant in artificial intelligence. The sensor values can be analyzed using AI and the health condition of the user can be classified [15]. As an example, the system can determine the readings whether normal, slightly abnormal or critical. In case of abnormal readings, the system can alert the user, caregiver or medical staff [16]. This renders the system more helpful as compared to plain sensor-based monitoring. The low-cost and compact design is another significant benefit of this project. The system is affordable with Raspberry Pi and readily available sensors, which are used in education and small-scale health care settings. The system can be experimented in a laboratory setting and subsequently enhanced to be used in the real world [17]. It also offers a powerful learning environment to students as it incorporates embedded systems, sensor interfacing, programming, data analysis, and decision-making based on AI.

2.Literature Review

Shalini K. S. et al. (2025) introduced a VLSI-based wearable health monitoring algorithm based on AI that emphasized predictive healthcare and energy-conscious hardware design. Their work demonstrated that the artificial intelligence can enhance the monitoring of the physiological conditions by recognizing the abnormal tendencies and facilitating early prediction [18]. Nevertheless, their work was more of an architecture of the system and concept of hardware direction and not a simple educational prototype which can be easily implemented and tested by students. Nandene et al. (2023) studied energy-efficient AI VLSI-based hardware and the significance of low-power processing in intelligent embedded systems [19]. Their input is valuable since health monitoring devices must be able to operate on minimum-energy usage particularly when they are applicable in wearable or portable-based settings. Nevertheless, their activity was primarily connected with the efficiency of hardware and did not deal directly with the practical combination with medical sensors. Huang et al. (2025) talked about AI-based wearable bioelectronics and how smart wearable systems can be utilized to aid in personalized digital healthcare [20]. Their research

highlighted the future of ongoing monitoring, electronic medical records and patient-specific analysis. Nevertheless, they also found privacy, interoperability, and data-sharing issues to be unresolved problems. Etli et al. (2024) conducted a review of AI-based wearable systems to predictive analytics and demonstrated that artificial intelligence could be used to assist in detecting anomalies, personalizing, and providing early warnings of diseases. This project is significant since their work can be used to transform raw sensor values into useful health decisions using AI. They also, however, observed that the availability of data sets, biasness, and model reliability are still significant issues. Kajornkasirat et al. (2025) investigated the concept of predictive analytics in wearable IoT-based health care systems and employed machine learning algorithms like the Random Forest to monitor health [21]. Their research proved that AI models could enhance the smart healthcare classification and prediction [22]. Nevertheless, their study was more general and was not completely oriented towards a student-level embedded implementation of Raspberry Pi. All these studies indicate that AI, wearables, edge intelligence, and low power hardware will soon be significant components of the modern healthcare monitoring [23]. Simultaneously, they show a definite gap in research and implementation. Numerous earlier papers have been written on the topic of advanced architectures, predictive models and future healthcare opportunities [24]. This gap justifies the current project which aims at an improved AI-based Raspberry Pi-based embedded system in health monitoring.

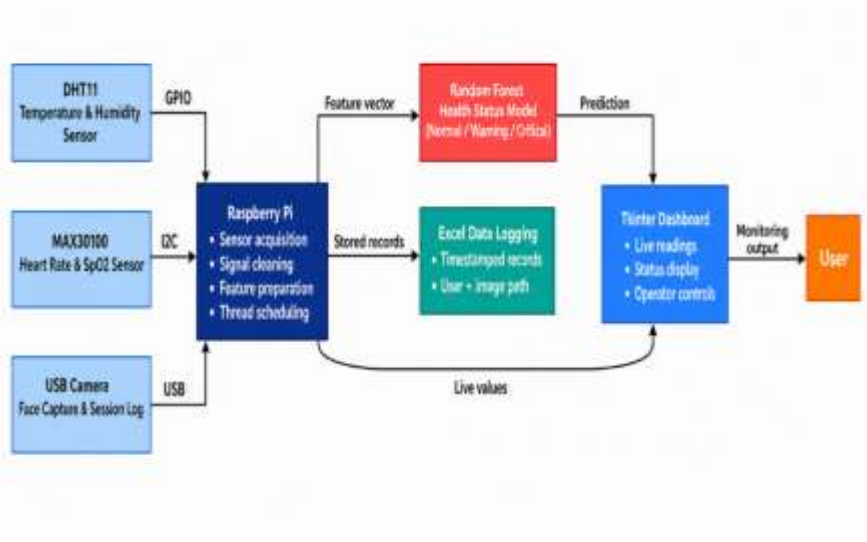
Kalaivani et al. (2025) surveyed the existing Raspberry Pi-based health monitoring systems and made Raspberry Pi a good inexpensive platform to use in healthcare [25]. Their activity is directly connected to the current study as Raspberry Pi offers the flexibility of software, compatibility with sensors, and sufficient processing capabilities to perform simple AI-based analysis. Their review, however, did not pay much attention to secure data integration, power optimization, and full implementation of the prototype at the level of a full prototype. The PMC healthcare monitoring review (2022) talked about the monitoring of patients using IoT and clarified the typical sensor-to-cloud flow that could be found in most healthcare systems. This review demonstrated how patient information could be gathered using sensors and sent to cloud platforms to be stored, analyzed and accessed remotely. This approach can be helpful but more relies on the internet connection and can be subjected to privacy threats due to sensitive health information. Conversely, the suggested Raspberry Pi-based system can facilitate local processing and thereby minimise cloud reliance as well as enhance response time. OVision et al. (2025) illustrated a medical device architecture on Raspberry Pi and demonstrated diagnostic capabilities of Raspberry Pi in the cheap healthcare setup. Their article demonstrated that Raspberry Pi can be implemented into medical-related systems, particularly where the low-cost computing is needed. Their interest was however primarily on the imaging of cancer diagnosis and not wearable physiological monitoring of things like heart rate, body temperature, oxygen level or other related vital signs. Shajari et al. (2023) examined AI-based wearable sensor technologies and provided a summary of the advances in digital health sensors in real-time healthcare monitoring [26]. Their analysis placed an emphasis on the utility of wearable sensors in gathering physiological data in real time, but also pointed out issues concerning real-time accuracy, signal quality, and power consumption. These issues are quite pertinent since realistic embedded health monitoring system should be dependable, responsive, and be energy-efficient. Ghadi et al. (2025) analyzed the importance of wearables and artificial intelligence in remote care and explained how digital healthcare systems can enhance the efficiency of monitoring, supporting patients, and making medical decisions [27]. They also focused on their work on the significance of interoperability, privacy protection, and standardization. But, as they observed, barriers to adoption and privacy issues continue to hinder the more general adoption of such systems. In general, the literature review indicates that the recent studies are highly in favor of applying AI, IoT, embedded platforms, wearable sensors, and Raspberry Pi in healthcare monitoring. Nevertheless, the vast majority of works are dealing with high-level concepts, advanced hardware design, or cloud-based healthcare flow, or specific diagnostic applications. It is still necessary to have a low-cost, practical, and easy to understand embedded prototype that integrates sensor data collection, Raspberry Pi processing, AI-based health condition prediction, and local decision-making. Thus, the current project is aimed to fill this gap by creating a more powerful AI-based Raspberry Pi-based embedded system capable of tracking health parameters, processing data in real-time, minimizing the network reliance, and enabling the timely health notification.

3. Methodology

This chapter outlines the research strategy that will be used in developing the Enhanced AI-Driven Raspberry Pi-Based Embedded System in Health Monitoring. The methodology is a description of how the proposed system was planned, simulated, designed, implemented, and tested as a realistic embedded health monitoring prototype. The primary aim of the methodology was to transform the theoretical concept

of AI-based medical monitoring into a practical system that would be able to gather physiological data, analyze the data on-site, estimate the health condition of the user, present the findings, and archive the measurements to be used later. The project methodology was developed in a step by step way in which, each step could support the other stage. It began by reviewing the literature and planning the system, hardware selection, software development, training the machine learning model, integrating the sensors, creating the dashboard, and finally, data logging. The uploaded project document details the final system is a combination of sensing, local processing, machine learning prediction and user interaction all in one embedded workflow.

Figure 2. System architecture of the implemented prototype



3.1 Research Design

The study design was an applied and experimental design. Given the fact that the objective of the project was to not just study the known health monitoring systems but also to create a working prototype, an implementation-based approach was chosen. The aim of the project was to create a low-cost embedded system with Raspberry Pi as the central controller. The system was to be used to track health parameters like temperature, humidity, heart rate and oxygen saturation level. These values were entered into a machine learning model as input features to predict the health status of the user as either Normal, Warning, or Critical. The research design was practical as it entailed direct interfacing of hardware and testing of software. Theoretical analysis was not the sole methodology used. Rather, the system was constructed, monitored and enhanced by trial and error. This method was appropriate in a student level embedded project since it explicitly showed the entire way to go through the concept to prototype. The chosen design also enabled local processing, i.e., the sensor values obtained were processed on the Raspberry Pi rather than being wholly reliant on cloud services. This rendered the system quicker, more convenient and more lab demonstration-appropriate.

3.2 Hardware Methodology

The hardware approach relied on the choice of low-priced, readily accessible and interoperable components. As the main embedded platform, the Raspberry Pi was employed. It was the primary controller of the entire system. Its tasks were to read sensor values, operate the graphical dashboard, load the trained machine learning model, predict health status, take images of the user and save data records. The choice of Raspberry Pi is based on the fact that it is capable of Python programming and can be easily combined with various libraries and hardware modules. Temperature and humidity were measured with DHT11 sensor. The humidity is not taken as a direct medical vital sign in the professional medical system, but in this project, it was involved as an environmental context variable. It also assisted in showing multi-parameter sensing in the system. The pulse-related data and oxygen saturation level were measured with the help of the MAX30100 sensor. This sensor was significant as the heart rate and SpO2 are directly related to the health monitoring and were utilized in the machine learning prediction process. The system was also equipped with a USB camera. Lacking medical diagnosis by camera. Its aim was to take the picture of the user at the beginning of the monitoring session and record the image path along with the sensor data. This helped in organising and identifying the monitoring session.

Table 1. Hardware components used in the final project

Component	Purpose in the System	Remarks
Raspberry Pi	Main controller, dashboard host, local AI inference	Flexible platform for Python-based embedded prototyping
DHT11	Temperature and humidity sensing	Simple and low-cost environmental sensor
MAX30100	Heart rate and SpO2 sensing	Digital pulse oximeter module used in the final prototype
USB Camera	Face image capture for session tagging	Used only for session identification support
Perfboard and enclosure	Practical mounting and wire management	Observed in hardware photographs
Power cable / USB supply	System power input	Suitable for laboratory demonstration setup

Table 2. Pin and interface mapping in the final prototype

Signal / Module	Interface Type	Implementation Detail
DHT11 data line	GPIO	Mapped in code through board.D4 on Raspberry Pi
MAX30100 SDA / SCL	I2C	Handled through the custom Python sensor driver
USB camera	USB	Opened through OpenCV for face capture
Machine learning model	Local file access	Loaded from health_model.pkl and label_encoder.pkl
Excel log storage	Filesystem	Saved inside sensor_data folder with user timestamp

Raspberry Pi was linked to the hardware via the relevant interfaces. The DHT11 data line was wired to a GPIO pin. The sensor (MAX30100) used the I2C interface. The USB camera was plugged in the USB port and opened with OpenCV. The installation configuration of the system was by practical wiring, perfboard support and enclosure. This assisted in enhancing the physical structure of the prototype and made it demonstrable.

3.3 Software Methodology

Python was used to develop the software methodology since Python allows interfacing with sensors, the creation of a graphical user interface, machine learning, data processing and image processing with just a single programming language. The software was created as a primary dashboard application, sensor reading functions, the MAX30100 driver, machine learning training script, and a data logging module. Most of the runtime operations of the system were under the control of the main application. The graphical dashboard was made using Tkinter. The dashboard enabled the user to input essential data with an option of initiating the monitoring session, see real-time sensor data, and monitor the predicted health status. The capture of the face image of the user was done using the USB camera with the help of OpenCV. Pandas was utilized to store and export sensor readings in an excel compatible format. Numpy was utilized in preparing numerical features in inputs of the machine learning model. The saved machine learning model and label encoder were loaded at run-time using Joblib. The program was built on cycles of repeated acquisitions. The values were constantly measured by the sensors and processed by the Raspberry Pi. Sensor reading was done separately in different threads to ensure that the graphical interface was responsive to monitoring. These values were stored temporarily, displayed on the dashboard and sent to the trained model and stored in the data file. This software methodology assisted in the integration of various components of the system into a single working embedded application.

3.4 Machine Learning Methodology

The machine learning process was created to be small, lightweight, and able to run on Raspberry Pi inference locally. The type of classifier that was chosen to predict the health status is a Random Forest classifier. The reason behind the choice of this model is that it is effective with structured tabular data, and it does not need as much computation as deep learning models, and it is simpler to train and interpret. The project used a dataset with four input features: Temperature, Humidity, HeartRate and SpO2. HealthStatus was the output variable and had three potential classes that include Normal, Warning and Critical. The training was conducted independently of the running application. The dataset was loaded into a CSV file. The health status labels were then coded into numerical form with the help of a label encoder. Then, the

data set was separated into the training and testing sets. Random Forest model was trained on the chosen input features. The model was trained using 150 estimators in the available project structure. Once trained, the trained model and the label encoder were stored as two different files. The Raspberry Pi application then loaded these saved files in real-time monitoring. The system gathered live sensor data during the runtime and packaged it into a four-feature input to the system. This input was fed to the trained Random Forest model. The model forecasted the health status of the user then. It was then converted back into a readable class label and shown on the dashboard. The methodology enabled the system to conduct local AI-based prediction without transmitting information to a cloud server.

3.5 Operational Methodology

The working methodology refers to how the system will work during a monitoring session. To begin with, the user boots up the application in the Raspberry Pi. The system prompt requests the user to fill in his or her name or some basic details. Following this, the USB camera will capture the image of the face of the user in case the camera is provided. The snapshot is stored and associated with the session information. Next, the system starts the DHT11 and MAX30100 sensors. The monitoring cycle starts as soon as the sensors are started. The DHT11 sensor measures the temperature and humidity levels, whereas the MAX30100 sensor measures pulse-related and SpO2 levels. The system verifies the validity and stability of sensor values. In case there are missing or unstable values, the system will suppress or disregard them until there are the proper values. The system prepares the input by the machine learning model when all the necessary parameters are available. Prediction of the health condition of the user is then predicted by the model. The outcome is presented on the dashboard through clear status indicators like Normal, Warning or critical. It can also be made easier to comprehend by color coding the result. As an illustration, a safe color can be used to indicate a normal condition whereas a greater alert color can be used to indicate a warning or critical condition. The readings are stored including the time of reading, user, sensor, prediction outcome, and image location. The process of monitoring goes on until the user ends the session or shuts the application.

3.6 Data Logging and Record Management

The aspect of data logging was significant to the methodology since a health monitoring system is not only expected to display live readings but also to store it to be accessed later. In this project, the sensor data were stored as individual sensor readings along with corresponding data like the time when it was recorded, the user name, sensor data, the estimated health condition, and the path to the face image. The data that had been logged were stored in an Excel compatible format within a special folder. This type of logging made the system easier to use in the academic demonstration and future analysis. It enabled the researcher to look at the values gathered once the session was completed. It also assisted in recording the behavior of the system in testing. The data was readily open, read and analyzed without using intricate database software by using Excel compatible storage. This was a workable and useful solution with regards to a prototype-level system.

3.7 Testing and Validation Methodology

The testing approach was to determine if the hardware, software and machine learning pieces fit together. Initially, the sensors were tested in isolation to determine if values were able to be read from the Raspberry Pi. The DHT11 sensor was tested for temperature and humidity. The MAX30100 sensor was tested for pulse and SpO2 values. The USB web camera was tested to make sure that images could be taken with OpenCV. Following the unit tests, the system was tested as a whole. During this testing, the sensors, dashboard, prediction model and logger were tested as an integrated application. We wanted to see whether the real-time readings were displayed on the dashboard, whether a health status was generated by the prediction model, and whether the data was saved to the file. This was critical because issues may only be detected when various components are integrated. For instance, sensor delays, sensor fluctuations, GUI hangs and file writing issues can only be detected at this stage.

3.8 Ethical and Practical Considerations

The system was developed for research purposes and not as a medical device. So, it should not be used as a medical diagnostic tool. The AI prediction was only applied to illustrate the classification of health status based on the sensor data and the data set. Clinical application would require medical sensors, larger datasets, clinical studies, safety studies and regulatory approval. Ethical considerations needed to be applied to the use of a camera. For the project, the camera was not used for facial diagnosis or biometric authentication; it was only used for session identification. The path of the image was saved together with

the monitoring data for demonstration. Health and identity data are sensitive, so it's crucial that in future iterations of the system privacy, security, and consent are enhanced.

4. Implementation And Experimental Setup

In this chapter, we discuss the implementation and experiments of the proposed Enhanced AI-Driven Raspberry Pi-Based Embedded System for Health Monitoring. The key aim of this phase was to implement the proposed system to create a prototype of an integrated system that collects health data, processes them locally, predicts the user's health status, displays the results, and logs the readings for future analysis. The proposed system was implemented using Raspberry Pi as the primary embedded system controller, along with biomedical and environmental sensors, a graphical user interface (GUI), machine learning model, USB camera and local data storage. The system was developed following a realistic approach, starting with simulation and progressing to hardware and software implementation. This ensured that the final system was not just a theoretical one but also practical and testable in the lab. The project report (see upload) indicates that the final implementation consisted of hardware simulation, prototype assembly, software modules, Tkinter dashboard, sensor acquisition, face capture, Excel logging and checkpoint testing.

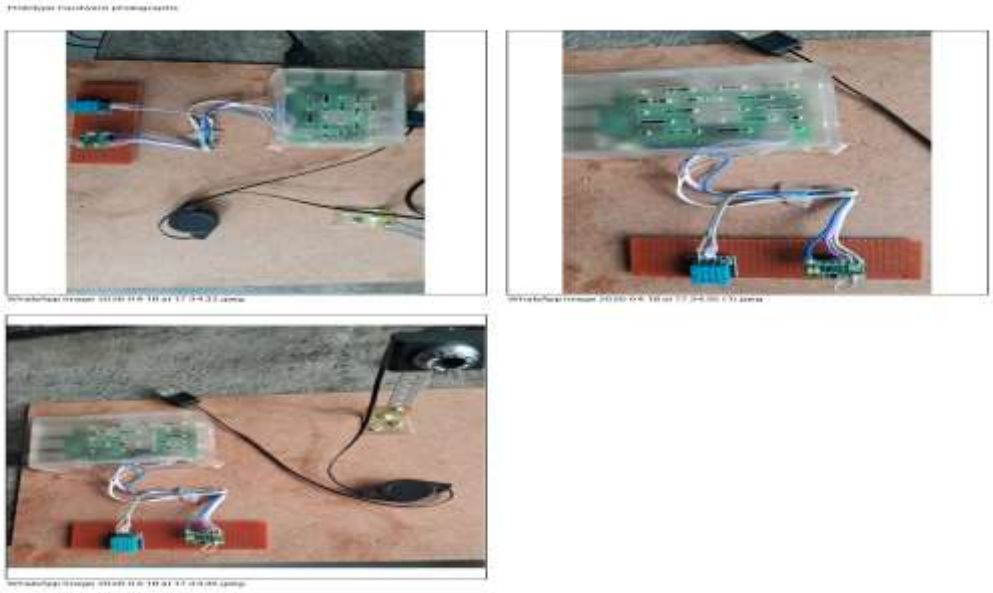
4.1 Hardware Simulation Stage

The initial interfacing of the system was tested in simulation before building the prototype. The initial hardware connection was tested using Proteus software. In the simulation stage, the Raspberry Pi 3 was connected with a DHT11 temperature and humidity sensor, an analog heartbeat sensor, an MCP3208 analog to digital converter and an LCD display. This stage was done to get an idea of how to connect various components and display health-related values in a basic embedded system. This preliminary simulation stage was necessary to reduce the risk of cause and effect before the physical stage. It ensured understanding of the flow of information from sensors to the Raspberry Pi. It also assisted in understanding the wiring, data flow and display. In a student embedded design project, this was helpful as the design could be verified before wiring. But the simulation also demonstrated that a design that may appear straightforward on a schematic is not necessarily the best option for a prototype. In particular, the analog pulse sensor required an ADC (analogue-to-digital converter) module. As a result, in the implementation, the analog pulse-sensing pathway was eliminated and a MAX30100 digital pulse oximeter sensor was used.

4.2 Final Prototype Assembly

The final hardware prototype consisted of Raspberry Pi single board computer, DHT11 temperature and humidity sensor, MAX30100 pulse oximeter sensor, USB camera, jumper wires, perfboard, power supply, and a basic enclosure. Raspberry Pi was the main computing unit of the system. It managed the sensor, dashboard, machine learning prediction, capturing and storing face images.

Figure 4. Prototype hardware photographs used for the final implementation chapter



The DHT11 sensor was used for collecting temperature and humidity readings. The MAX30100 sensor was used to collect pulse-related and oxygen saturation. The USB camera was used to take a picture of the user's

face at the start of the session to identify the session. This was a prototype for the classroom setting and not for commercial use. Hence, the design emphasized functional convenience, visibility and accessibility. The perfboard and enclosure facilitated hardware organization and improved the appearance of the system. During the assembly, we encountered problems with loose connections, sensor placement, inaccurate sensor readings, and wiring. These are typical problems in designing embedded systems and are resolved during the implementation.

4.3 Software Implementation

The system software was written in Python as this programming language supports interfacing with hardware, designing graphical user interface, running machine learning algorithms, image processing and file operations. The final software design was separated into modules. The core dashboard program managed the entire monitoring process. A separate driver file was used to communicate with the MAX30100 sensor. A training file was implemented to train the machine learning model, and a test file to test the MAX30100 sensor.

Table 3. Summary of source files used in the final implementation

File Name	Role in the Project	Implementation Notes
ME_health_monitor2.py	Main dashboard application	Handles user entry, face capture, sensors, prediction, dashboard update, and Excel logging
max30100.py	Sensor interface driver	Provides low-level access to MAX30100 registers and reading functions
health_training.py	Machine learning training	Trains Random Forest classifier and saves model files
testMAX30100.py	Quick module verification	Simple script used for checking pulse and SpO2 values from the sensor

The main file of the application managed user input, face detection, sensor setup, prediction, updating the dashboard and logging to Excel. This code architecture ensured the project was easier to comprehend, test and present. Rather than having everything in one code file, the implementation separated the communication with the sensors, training the model, and monitoring the system during operation. This made the system easier to understand and facilitates writing of the final report.

4.4 Tkinter Dashboard Implementation

A Tkinter-based dashboard was created to run the live monitoring. This dashboard was used instead of the LCD screen in the simulation. This dashboard offered a bigger screen for displaying temperature, humidity, heart rate, SpO2, raw IR value, sensor status and estimated health status. It also had simple buttons for starting, stopping, clearing and saving of data. This dashboard was created to enhance the user experience. It not only showed the numbers, but also the predicted condition as Normal, Warning or Critical. The status was also represented using colours. This was necessary because people might not necessarily understand the meaning of sensor readings. Such a status label allows them to easily check if the status looks normal or critical.

Figure 5. Live dashboard snapshot showing temperature, humidity, heart rate, SpO2, and predicted status

4.5 Sensor Acquisition Logic

The sensor acquisition code was designed to acquire data from the sensors. The DHT11 sensor was used to collect temperature and humidity at regular intervals. These were then sent to the dashboard via a queue. This allowed the interface to remain responsive while the sensors were being read. The MAX30100 sensor was used to gather pulse data and SpO2. There is occasionally instability in the values provided by low-cost sensors, so the code implemented simple checks. The system validated the data were within normal ranges before it presented the data and used it for prediction. The dashboard did not consider data as valid if they were missing or unstable. This enhanced the prototype's real-world performance. The implementation using threads and queues improved performance. Reading sensor data was done in the background while the GUI was running. The dashboard did not get stuck while visualising the predictions. So the implementation went beyond polling the sensors, imbuing the system with a process (at run-time).

4.6 Machine Learning Integration

The system was incorporated into the Raspberry Pi-based system, with its machine learning model that classifies the health condition of the user. The reason as to why a Random Forest classifier was employed is that this type of classifier is appropriate when dealing with structured numerical data, and does not demand high computation needs. Four features of input were utilized in the model temperature, humidity, heart rate, and SpO2. According to these values, the model was able to predict the health condition as Normal, Warning, or Critical. It was done separately and with the available dataset to train. The encoder model and the label encoder were saved as files after training. These saved files were loaded during run time and were used by the main application in predicting. Such a division of training and deployment simplified the end-system, and made it easier and more realistic. Training the model required with the Raspberry Pi was not necessary every time. It just took the trained model and applied it in real time inference.

4.7 Face Capture and Session Tagging

The system was also equipped with a USB camera to take a picture of the face of the user at the start of the session. This feature was meant to do neither medical diagnosis nor to biometrically authenticate. It was simply applied to tagging of sessions. Once the user began the application, the system asked the user his/her name and tried to take a face photo. The image captured was stored as a time-stamped file name and attached to the sensor data sent. This aspect facilitated improved organization of the monitoring session. It enabled the data saved to be linked to a particular user session. In case the camera did not get an image, the system was able to enable the user to continue monitoring in case he wished to do so. This enabled the prototype to be more durable and avoid the overall application halting because of a problem with the camera.

4.8 Data Logging and Local Storage

Pandas was used to do data logging. The system stored readings in an excel compatible file that has a time-stamped name. The details that were of great importance in each entry that was logged consisted of the following: timestamp, user name, sensor type, predicted health state, face image path and sensor values. This rendered the system more practical in that it was able not only to give a live reading, but also to save this data which could be reviewed subsequently. Local storage was chosen due to its simplicity and also it is the one that is appropriate to a student prototype. It also minimizes the use of access to the internet. The Excel files saved can be opened in the future to analyze, prepare reports or compare various test sessions. This functionality enhances the usefulness of the system and demonstrates the prototype took into consideration data persistence.

4.9 Experimental Procedure

The experimental system had an easy and reproducible routine. Initially, the Raspberry Pi was booted and dashboard application was opened. The user typed in his name, and it tried to capture faces using the USB camera. Separately, DHT11 and MAX30100 sensors were set up. As soon as the sensors turned on, live dashboard began to show readings. The observations of the readings were repeated with successive acquisition cycles. Humidity, temperature, pulse rate and the SpO2 were observed on the dashboard. These values were fed into the machine learning model which showed the predicted health condition. To capture the system behavior, screenshots were recorded when undergoing various experiments. Trials indicated valid readings and predicted status in some cases and this was missing or unstable pulse and SpO2 values in others. The observations allowed to assess the working ability of the prototype.

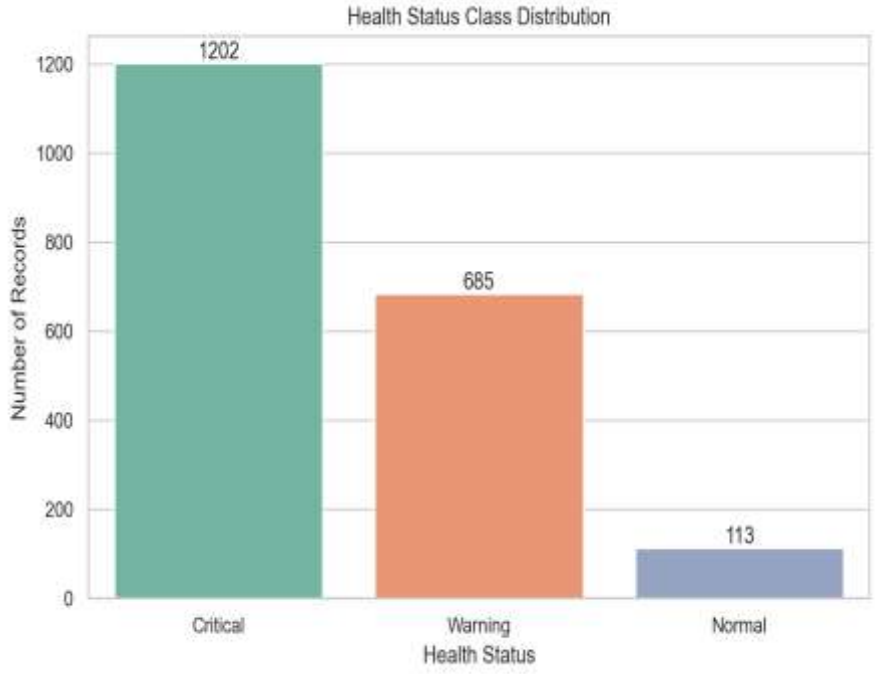
4.10 Implementation Challenges

There were a number of issues witnessed throughout implementation. The first one is the gap between the way the architecture was to be and the system that was actually implemented. Planning in the past also involved more other advanced options like cloud dashboards, secure communication and TensorFlow Lite optimization. The last system was however based on a local Tkinter dashboard and Random Forest model. This choice made this project more realistic and phenomena within the resources offered. Sensor stability was another problem. The sensor (MAX30100) at times gave erratic or no readings. This occurs in cheap sensor modules and is dependent on the location, contact quality, and their environment. This problem was dealt with in the system through the use of validation logic along with recording the state of valid and invalid reading. The third problem was that model performance had to be interpreted. The dataset delivered extremely robust outcomes, which, however, should be approached cautiously since controlled datasets are not likely to reflect actual clinical conditions.

5. Results And Discussion

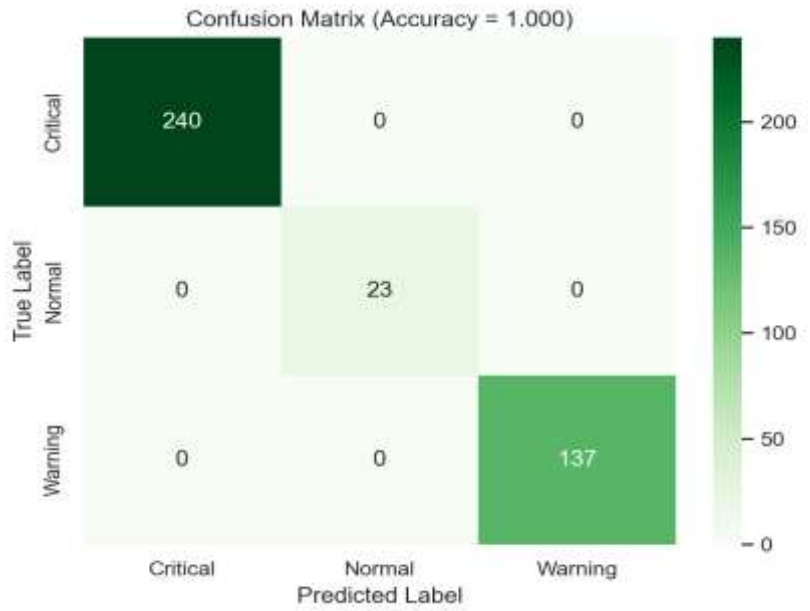
Findings of the Enhanced AI-Driven Raspberry Pi-Based Embedded System on Health Monitoring indicate that this system was effectively utilized as both a prediction model as well as real-time monitoring platform. The analysis was conducted via dataset based model testing, live dashboard watching and practical implementation analysis. The model was utilized in the classification of health conditions in sub-sets based on the available local data, whereas monitoring sensor values in the dashboard were easily readable in repeated runsets. Live readings showed differences in response to the changing health parameters in the system. On balance, the project proved to be well usable with an appropriate data logging and strong hardware, software and AI-based health monitoring integration.

Figure 6. Class distribution of the available health dataset



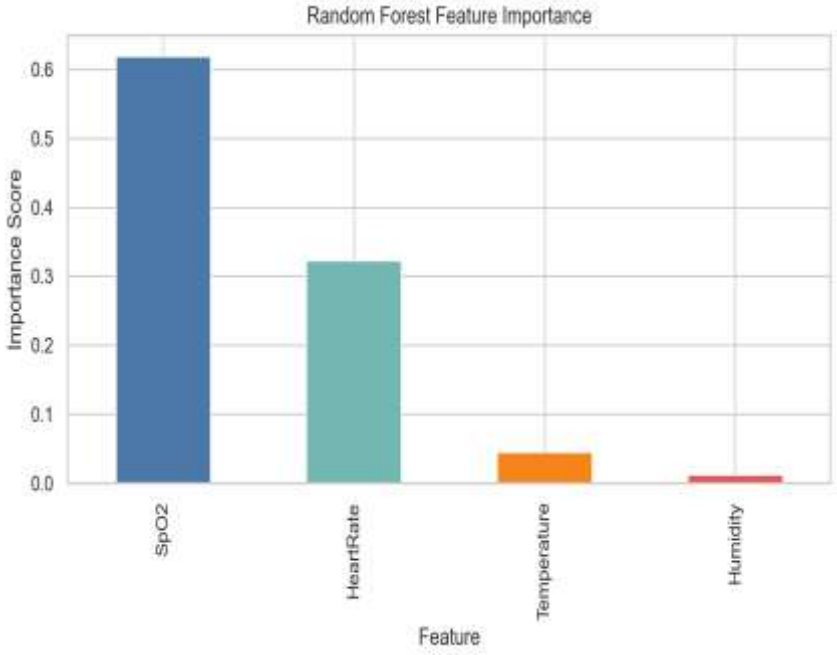
Bar chart shows the distribution of classes of health status in a dataset. The highest records are found in the "Critical" health status category, which includes 1202 records, which represents a considerable share of the data. The category of Warning comes second with 685 records; it is observed to be of moderate presence. Conversely, the health condition of the class of "Normal" health status has the fewest number of records as it includes 113 records. This distribution shows that there is an unbalanced dataset with more critical and warning statuses compared to normal health status, which might need to be addressed either to analyze it properly or conduct additional data analysis.

Figure 7. Confusion matrix obtained from the retrained Random Forest model



The confusion matrix shows that the model is perfectly accurate, i.e., a score of 1.0. In the case of the critical status of health, the model was able to accurately classify 240 cases and none was misclassified. In a similar way, when it came to the "Normal" class, there were 23 instances correctly predicted, and no mistakes were made. In the class Warning, there were 137 correct predictions. Notably, no false positives or false negatives (that all the predicted labels exactly matched the true labels). This outcome underscores the effectiveness of this model in categorizing the health status categories with no misclassification having a perfect performance on all the categories.

Figure 8. Feature importance values learned by the Random Forest classifier



The bar chart represents the scores of feature importance of a Rand Forest model. SpO2 is the most significant among the features with an importance score of around 0.6. HeartRate will come next with a score of approximately 0.3 meaning that it has a medium influence on predictions of the model. The indicators of Temperature and Humidity have very low significance and their scores are near to 0.1 or below which implies that the characteristics play a very weak role in the overall determination of the model. This emphasizes that, SpO2 is the leading characteristic, which has a great influence in the performance of the model.

Figure 9. Correlation between the sensor features in the available dataset

The correlation heatmap shows all the connections among the sensor characteristics. The diagonal values (all 1.0) indicate the ideal correlation of each feature with itself. Off-diagonal values indicate the relationships between various features. Temperature feature has the weak associations with Humidity (-0.02), HeartRate (0.01), and SpO2 (-0.03). Likewise, HeartRate (-0.02) and SpO2 (-0.03) are weakly correlated with Humidity. Both HeartRate shows a near-zero correlation with SpO2, thus no significant linear relationship. In general, the weak relationships indicate that the characteristics are mostly unrelated, and they provide different information to the model.

Figure 10. Dashboard screenshots captured during multiple monitoring trials

The painting consists of a series of snapshots of a dashboard in the case of monitoring experiments. Key health metrics, such as Temperature (°C), Heart Rate (BPM), and SpO2 (%,) are shown on each snapshot. These measures are observed at various points in time with different values in different trials. To illustrate, in the upper-left section, the initial snapshot displays Temperature: 34.3C, Heart Rate: 58BPM, and SpO2: 79, whereas other snapshots indicate alternative values, as Temperature increases or decreases to 34.2C - 34.5C, and Heart Rate changes to 58-78BPM. The SpO2 also fluctuate decently to indicate the varying conditions that were observed in the trials. The screenshots appear to show the real-time tracking and monitoring of these health indicators.

6. Conclusion

In the project, a full student level prototype of AI-based health monitoring to use Raspberry Pi was developed and reported. As the work has shown, Raspberry Pi could also be a helpful embedded platform when it comes to experiments involving health monitoring, and flexibility and rapid development are valued more than end-of-the-world hardware efficiency. The project demonstrates also that a lightweight machine learning model can be embedded into the workflow of the runtime without having to process externally in the cloud. However, most of all, the project helped me to understand that an ideal report should not simply reflect ideal expectations in a final report. It must also present a true design trail, modifications in implementation, realistic sensor action, and sincere analysis of the outcomes. To this end, this report records the strengths in addition to the current limitations in the prototype.

7. Future Scope

The initial and the most essential future change is improved data collection. The machine learning assessment would be more robust with a larger, more lifelike, and more balanced dataset, and would enable me to verify the hypothesis of whether the model presently generalizes across users and situations. This ought to involve repeat assessments by different persons, varying body conditions, and at variances of environmental conditions. Communication and security improvement is the second direction of the future. The Phase 1 project proposal touched on encryption and monitoring based on the network and the final code was only a local prototype. One can include a secure wireless transmission, lightweight remote dashboard, user authentication and preserve privacy in a future version. The third direction of the future is the hardware refinement. Stasis, roomier enclosure, cleaner board layout, and better biomedical sensors would enhance the quality of signals and usability. When the project is taken to the greater VLSI and embedded optimization direction that has been noted in the title and the literature review, then the implementation of the Raspberry Pi can be the software reference model wherein further specialization of hardware can be done in the future. The fourth future improvement is improving the model. The health status output can be more informative with more sophisticated models confidence estimation, anomaly explanation, and temporal trend analysis. Nonetheless, these additions can be constructed only once the robust data quality and sensor reliability is determined.

8. References

[1] D. T. Liss, T. Uchida, C. L. Wilkes, A. Radakrishnan, and J. A. Linder, “General Health Checks in Adult Primary Care: A Review,” JAMA, vol. 325, no. 22, p. 2294, Jun. 2021, doi: 10.1001/jama.2021.6524.

[2] D. Dias and J. Paulo Silva Cunha, “Wearable Health Devices—Vital Sign Monitoring, Systems and Technologies,” Sensors, vol. 18, no. 8, p. 2414, Jul. 2018, doi: 10.3390/s18082414.

[3] N. El-Rashidy, S. El-Sappagh, S. Islam, H. M. El-Bakry, and S. Abdelrazek, “Mobile Health in Remote Patient Monitoring for Chronic Diseases: Principles, Trends, and Challenges,” Diagnostics, vol. 11, no. 4, p. 607, Mar. 2021, doi: 10.3390/diagnostics11040607.

[4] V. Pardeshi, S. Sagar, S. Murmurwar, and P. Hage, “Health monitoring systems using IoT and Raspberry Pi — A review,” in 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bengaluru, India: IEEE, Feb. 2017, pp. 134–137. doi: 10.1109/ICIMIA.2017.7975587.

[5] E. Bourke-Matas, E. Bosley, K. Smith, B. Meadley, and K.-A. Bowles, “Developing a consensus-based definition of out-of-hospital clinical deterioration: A Delphi study,” Australian Critical Care, vol. 37, no. 2, pp. 318–325, Mar. 2024, doi: 10.1016/j.aucc.2023.05.008.

[6] R. M. D. D. Rathnayake, A. M. P. R. B. Arawa, and R. M. T. C. B. Ekanayake, “AI in Wearable Embedded Systems for Healthcare Monitoring: A Review,” Sri Lankan J. App. Sci., vol. 4, no. 1, pp. 41–54, Aug. 2025, doi: 10.4038/sljoas.v4i1.15.

[7] C. Virginia Anikwe et al., "Mobile and wearable sensors for data-driven health monitoring system: State-of-the-art and future prospect," Expert Systems with Applications, vol. 202, p. 117362, Sep. 2022, doi: 10.1016/j.eswa.2022.117362.
[8] C.-Y. Cheng, P. Pourhejazy, C.-Y. Hung, and C. Yuangyai, "Smart Monitoring of Manufacturing Systems for Automated Decision-Making: A Multi-Method Framework," Sensors, vol. 21, no. 20, p. 6860, Oct. 2021, doi: 10.3390/s21206860.
[9] E. Al-Masri, S. Kabu, and P. Dixith, "Emerging Hardware Prototyping Technologies as Tools for Learning," IEEE Access, vol. 8, pp. 80207–80217, 2020, doi: 10.1109/ACCESS.2020.2991014.
[10] P. Verma and S. Fatima, "Smart Healthcare Applications and Real-Time Analytics Through Edge Computing," in Internet of Things Use Cases for the Healthcare Industry, P. Raj, J. M. Chatterjee, A. Kumar, and B. Balamurugan, Eds., Cham: Springer International Publishing, 2020, pp. 241–270. doi: 10.1007/978-3-030-37526-3_11.
[11] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, and N. K. Kumaravelu, "A comprehensive review on applications of Raspberry Pi," Computer Science Review, vol. 52, p. 100636, May 2024, doi: 10.1016/j.cosrev.2024.100636.
[12] P. A. Pankov, I. V. Nikiforov, and D. F. Drobintsev, "Hardware and software data processing system for research and scientific purposes based on Raspberry Pi 3 microcomputer," Proceedings of ISP RAS, vol. 32, no. 3, pp. 57–69, 2020, doi: 10.15514/ISPRAS-2020-32(3)-5.
[13] M. H. Bakri, A. C. Özarslan, A. Erarslan, Y. Basaran Elalmis, and F. Ciftci, "Biomedical applications of wearable biosensors," Next Materials, vol. 3, p. 100084, Apr. 2024, doi: 10.1016/j.nxmate.2023.100084.
[14] A. Hassan et al., "Performance Evaluation of Raspberry Pi as an IoT Edge Signal Processing Device for a Real-time Flash Flood Forecasting System," IJACSA, vol. 13, no. 10, 2022, doi: 10.14569/IJACSA.2022.01310100.
[15] S. B. Junaid et al., "Artificial Intelligence, Sensors and Vital Health Signs: A Review," Applied Sciences, vol. 12, no. 22, p. 11475, Nov. 2022, doi: 10.3390/app122211475.
[16] T. R. Aditya, S. S. Pai, K. Bhat U, P. Manjunath, and G. Jagadamba, "Real Time Patient Activity Monitoring and Alert System," in 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India: IEEE, Jul. 2020, pp. 708–712. doi: 10.1109/ICESC48915.2020.9155602.
[17] J. Álvarez Ariza and C. Nomesqui Galvis, "RaspyControl Lab: A fully open-source and real-time remote laboratory for education in automatic control systems using Raspberry Pi and Python," HardwareX, vol. 13, p. e00396, Mar. 2023, doi: 10.1016/j.ohx.2023.e00396.
[18] S. K S, A. M N, P. P S, M. G, M. S B, and N. H V, "AI-Driven VLSI Design for Wearable Health Monitoring Devices: Real-Time Analysis and Predictive Insights for Vital Signs," in 2025 International Conference on Recent Advances in Electrical, Electronics, Ubiquitous Communication, and Computational Intelligence (RAEEUCCI), Chennai, India: IEEE, Apr. 2025, pp. 1–6. doi: 10.1109/RAEEUCCI63961.2025.11048340.
[19] Nandene. D P, Nivitha. G, Priyadharshini. R, and Arunkumar. K, "Energy-Efficient VLSI Hardware for Edge AI in Image Processing," in 2023 International Conference on Sustainable Communication Networks and Application (ICSCNA), Theni, India: IEEE, Nov. 2023, pp. 1605–1611. doi: 10.1109/ICSCNA58489.2023.10370425.
[20] G. Huang, X. Chen, and C. Liao, "AI-Driven Wearable Bioelectronics in Digital Healthcare," Biosensors, vol. 15, no. 7, p. 410, Jun. 2025, doi: 10.3390/bios15070410.
[21] S. Kajornkasirat, C. Sawangwong, K. Puangsuwan, N. Chanapai, W. Phutthamongkhon, and S. Puttinaovarat, "Integrating AI-Driven Predictive Analytics in Wearable IoT for Real-Time Health Monitoring in Smart Healthcare Systems," Applied Sciences, vol. 15, no. 8, p. 4400, Apr. 2025, doi: 10.3390/app15084400.
[22] A. Al-Nafjan, A. Aljuhani, A. Alshebel, A. Alharbi, and A. Alshehri, "Artificial Intelligence in Predictive Healthcare: A Systematic Review," JCM, vol. 14, no. 19, p. 6752, Sep. 2025, doi: 10.3390/jcm14196752.
[23] S. Shajari, K. Kuruvinashetti, A. Komeili, and U. Sundararaj, "The Emergence of AI-Based Wearable Sensors for Digital Health Technology: A Review," Sensors, vol. 23, no. 23, p. 9498, Nov. 2023, doi: 10.3390/s23239498.
[24] D. Fernandes Prabhu, V. Gurupur, A. Stone, and E. Trader, "Integrating Artificial Intelligence, Electronic Health Records, and Wearables for Predictive, Patient-Centered Decision Support in Healthcare," Healthcare, vol. 13, no. 21, p. 2753, Oct. 2025, doi: 10.3390/healthcare13212753.

[25] P. Baraneedharan, S. Kalaivani, S. Vaishnavi, and K. Somasundaram, "Revolutionizing healthcare: A review on cutting-edge innovations in Raspberry Pi-powered health monitoring sensors," Computers in Biology and Medicine, vol. 190, p. 110109, May 2025, doi: 10.1016/j.compbiomed.2025.110109.
[26] S. Shajari, K. Kuruvinashetti, A. Komeili, and U. Sundararaj, "The Emergence of AI-Based Wearable Sensors for Digital Health Technology: A Review," Sensors, vol. 23, no. 23, p. 9498, Nov. 2023, doi: 10.3390/s23239498.
[27] Y. Y. Ghadi et al., "Integration of wearable technology and artificial intelligence in digital health for remote patient care," J Cloud Comp, vol. 14, no. 1, p. 39, Jul. 2025, doi: 10.1186/s13677-025-00759-4.