INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 07 ISSUE: 11 | NOVEMBER - 2023 SJIF RATING: 8.176 ISSN: 2582-3930

IoT Based Novel Approach for Remote Patient Pulse Rate Monitoring and Stroke Prediction using Machine Learning

Prajakta P. Musale, Arya M. Pathak, Pranav K. Patel, Rukmoddin N. Patel, Aarushi P. Pathak, Ameya R. Pathak, Revati B. Pathak

Abstract *—In recent years, the advancement of Internet of Things (IoT) technologies has revolutionized various domains, including healthcare. Remote patient monitoring is an emerging field that allows healthcare professionals to monitor vital signs of patients remotely, ensuring timely intervention and improved healthcare outcomes. This research paper presents a system that combines the ESP32 microcontroller, an Express MVC website, and a web API to enable remote pulse monitoring. The ESP32 collects pulse data and sends HTTP requests to a web API, which then responds with live data to be displayed on the Express MVC website. Additionally, based on specific response codes from the web API, the ESP32 triggers a WhatsApp bot to notify healthcare professionals. The proposed system demonstrates the feasibility and potential for remote pulse monitoring using readily available IoT components*

.

Keywords *— Remote patient monitoring, ESP32, IoT healthcare, Express MVC, MongoDB timeseries*

I. INTRODUCTION

1.1 Background

Remote patient monitoring has emerged as a vital component in the modern healthcare landscape. It enables healthcare professionals to remotely monitor patients' vital signs and health parameters, allowing for early detection of abnormalities and timely intervention. Traditionally, pulse monitoring required patients to be physically present in healthcare facilities. However, with the advent of Internet of Things (IoT) technologies, it is now possible to monitor pulse data remotely, providing convenience and improved healthcare outcomes.

Stroke is a leading cause of death and disability worldwide, necessitating the development of accurate prediction models to identify individuals at risk. In recent years, machine learning algorithms have gained prominence in healthcare research due to their ability to extract complex patterns and make predictions based on diverse sets of data. This paper aims to explore the application of machine learning techniques in Python for stroke prediction. By utilizing a dataset containing demographic, lifestyle, and medical information, we will train and evaluate various machine learning models to identify the most effective predictive model for stroke.

1.2 Problem Statement

While remote patient monitoring has gained significant attention, there are still challenges to be addressed. Existing systems often lack the necessary flexibility, accessibility, and real-time data visualization features. Moreover, seamless integration with popular communication channels for instant notifications to healthcare providers is also an area that requires attention. To bridge these gaps, there is a need for a robust remote pulse monitoring system that leverages IoT technologies and provides real-time data visualization, seamless data transmission, and instant notifications.

1.3 Objectives

The objective of this research paper is to develop a remote pulse monitoring system that addresses the limitations of existing systems. The system combines the capabilities of the ESP32 microcontroller, an Express MVC website, and a web API to enable efficient and effective remote pulse monitoring. The specific objectives of this research are as follows:

1. Develop the ESP32 implementation for pulse data collection and transmission.
2. Design and implement a web API to handle HTTP requests and provide live pulse data responses.
3. Create an Express MVC website for real-time data visualization and user interaction.
4. Integrate the ESP32, web API, and Express MVC website to establish a seamless end-to-end system.
5. Implement a mechanism to trigger a WhatsApp bot based on specific response codes from the web API, providing instant notifications to healthcare professionals.

By achieving these objectives, this research aims to contribute to the advancement of remote patient monitoring systems, enabling healthcare providers to remotely monitor patients' pulse data, visualize it in real-time, and receive timely notifications for critical situations.

II. METHODOLOGY/EXPERIMENTAL

A. Method

Sensor data plays a crucial role in the era of the Internet of Things (IoT), enabling us to gain insights and make informed decisions in various domains. However, understanding and analyzing sensor data can be challenging due to its sheer volume and complexity. In this article, we will explore how to leverage the Express MVC framework, MongoDB TimeSeries, and Highcharts JS library to visualize sensor data effectively and derive valuable insights.

Express MVC:
Express MVC is a popular web application framework built on top of Node.js. It follows the Model-View-Controller (MVC) architectural pattern, which helps separate the concerns of data manipulation, presentation logic, and user interaction. By utilizing Express MVC, we can build a robust and scalable sensor data visualization application.

MongoDB TimeSeries:
MongoDB TimeSeries is a specialized feature introduced in MongoDB 5.0 designed specifically for efficient storage and analysis of time-series data. Time-series data represents information collected over time, such as sensor readings. MongoDB TimeSeries provides native support for time-series data and offers optimized storage, retrieval, and aggregation capabilities. Leveraging MongoDB TimeSeries, we can store and query sensor data in a highly efficient manner, facilitating faster analysis and visualization.

Highcharts JS:
Highcharts JS is a powerful JavaScript charting library that provides a wide range of interactive and visually appealing charts. It offers support for various chart types such as line charts, area charts, and scatter plots, making it ideal for visualizing time-series data. Highcharts JS also provides extensive customization options, allowing us to tailor the charts to specific requirements and enhance the overall data visualization experience.

Setting Up the Environment:
To begin, we need to set up our development environment. Start by installing Node.js and MongoDB. Once installed, create a new Express MVC project using the Express application generator. Next, install the necessary dependencies, including the MongoDB TimeSeries plugin and the Highcharts JS library. These dependencies will provide the required functionality for handling sensor data and visualizing it effectively.

Data Collection and Storage:
In a sensor data visualization application, the first step is to collect and store the sensor data. This is typically achieved by connecting to the sensor devices using appropriate protocols such as MQTT or RESTful APIs. The collected data is then stored in MongoDB TimeSeries, leveraging its specialized storage capabilities for time-series data. By efficiently storing the data, we ensure easy retrieval and analysis in the subsequent steps.

Data Collection and Preprocessing:
To develop a reliable stroke prediction model, a comprehensive dataset containing relevant features is crucial. The dataset used in this study consists of demographic attributes (age, gender), lifestyle factors (smoking status, alcohol consumption), and medical information (hypertension, heart disease, etc.). Additionally, the dataset may contain information regarding BMI (Body Mass Index), glucose levels, and previous medical history. Careful consideration is given to ensure data quality, including handling missing values, handling categorical variables through encoding techniques, and normalizing numerical features to a consistent scale.

Data Retrieval and Aggregation:
Once the sensor data is stored in MongoDB TimeSeries, we can retrieve and aggregate the data for visualization. MongoDB TimeSeries provides powerful querying capabilities, allowing us to filter and retrieve specific data based on time ranges, sensor types, or other relevant parameters. Additionally, we can perform aggregations on the data, such as calculating averages, maximums, or minimums over specific time intervals. These aggregations can provide valuable insights and help identify patterns or anomalies in the sensor data.

Integration with Express MVC:
With the sensor data retrieved and aggregated, we can now integrate it into our Express MVC application. Create appropriate models to represent the sensor data, controllers to handle data retrieval and processing, and views to render the visualizations using Highcharts JS. Express MVC provides a structured and modular approach to manage the application's components, making it easier to maintain and extend the codebase.

Visualization with Highcharts JS:
Highcharts JS offers a rich set of options for visualizing sensor data. Depending on the nature of the data and the insights we want to convey, we can choose from various chart types, such as line charts, area charts, or scatter plots. We can customize the appearance of the charts by configuring axes, labels, colors, and tooltips. Highcharts JS also supports interactive features like zooming and panning, enabling users to explore the data in more detail.

Real-time Updates:
In many cases, sensor data is continuously generated in real-time. To provide a dynamic visualization experience, we can leverage technologies like WebSockets or server-sent events (SSE) to stream the data from the server to the client in real-time. Highcharts JS provides mechanisms to update the charts dynamically as new data arrives, ensuring that the visualizations reflect the latest sensor readings.

Feature Selection and Engineering:
Feature selection plays a vital role in constructing an accurate and interpretable model. Through exploratory data analysis, we gain insights into the relationships between features and their potential impact on stroke prediction. Various feature selection techniques, such as correlation analysis, mutual information, and recursive feature elimination, can be employed to identify the most influential features. Additionally, domain expertise and medical knowledge can guide the selection of relevant features, ensuring clinical relevance and interpretability.

Machine Learning Models:
In this research, we implement and compare several machine learning algorithms commonly used in binary classification tasks,

including logistic regression, random forest, and support vector machines (SVM). These models are trained on the preprocessed dataset, using appropriate machine learning libraries in Python, such as scikit-learn. During the training phase, the dataset is split into training and validation sets to assess model performance. Model hyperparameters are tuned using techniques such as grid search or random search to optimize performance.

FIGURE 1
SYSTEM APPROACH

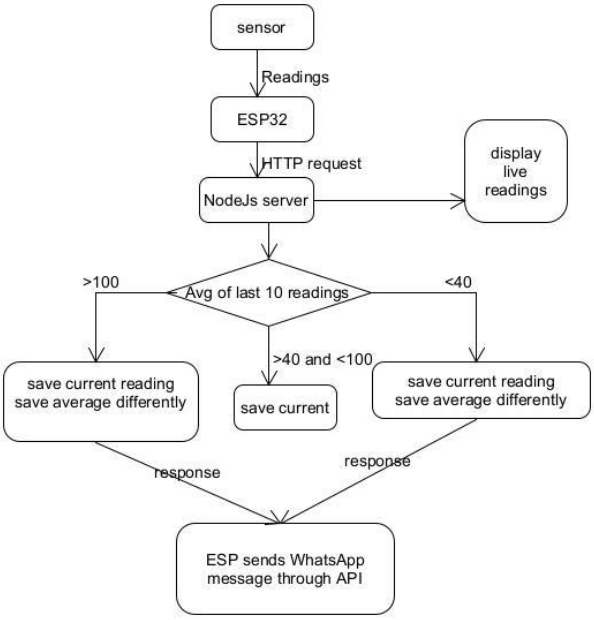


FIGURE 2
LOGISTIC REGRESSION

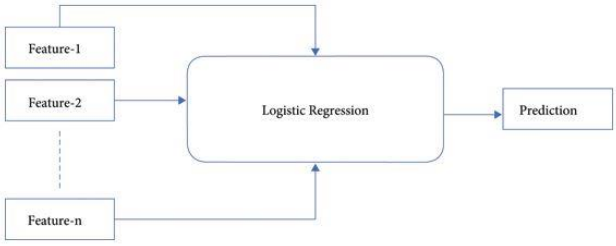
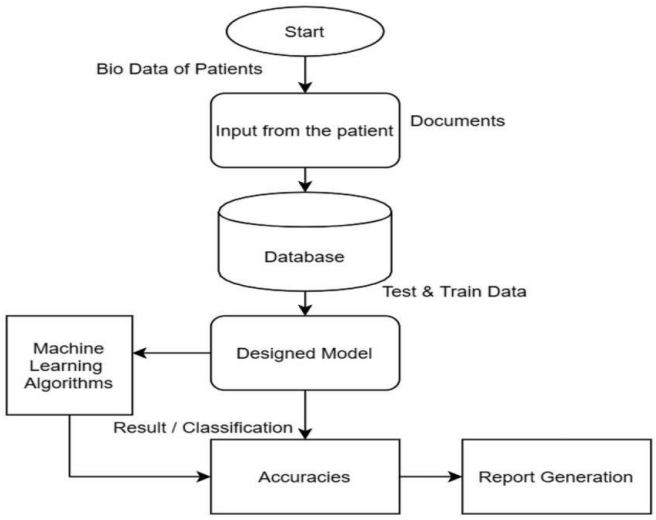


FIGURE 3
STROKE PREDICTION FLOWCHART



III. RESULTS AND DISCUSSIONS

Model Evaluation:

To evaluate the performance of the machine learning models, various evaluation metrics are employed, including accuracy, precision, recall, and F1-score. Additionally, the area under the receiver operating characteristic curve (AUC-ROC) is used as a measure of overall model performance. Cross-validation techniques, such as k-fold cross-validation, can be utilized to obtain more reliable performance estimates and mitigate overfitting issues. The models are assessed and compared based on their performance metrics to identify the most effective algorithm for stroke prediction.

FIGURE 3
PREDICTION RESULTS

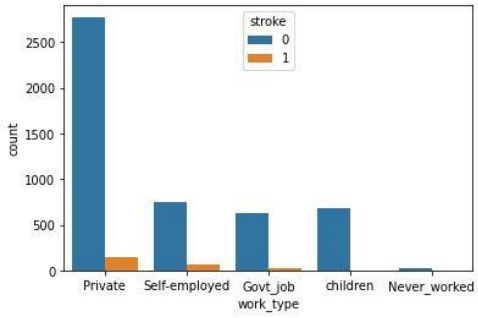


FIGURE 4
SHOWCASE OF LIVE PULSE MONITORING

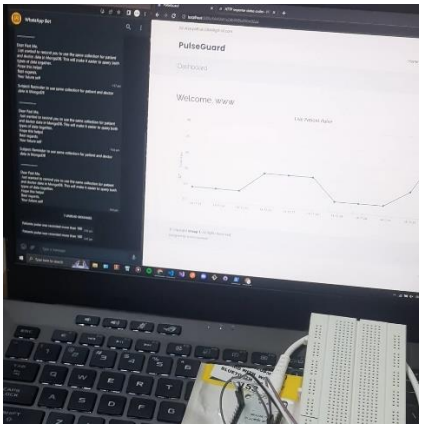


FIGURE 5
WHATSAPP BOT WORKING

IV. FUTURE SCOPE

1. Expansion to Multiple Vital Signs Monitoring: The system can be extended to monitor and transmit data for additional vital signs such as blood pressure, oxygen saturation levels, or respiratory rate. This would provide a more comprehensive remote patient monitoring solution, enabling healthcare providers to gather a wider range of vital sign data remotely.

2. Integration with Machine Learning Algorithms: By incorporating machine learning algorithms, the system can be enhanced to detect patterns, anomalies, or early warning signs in the pulse data. This would enable automated analysis and alert generation for critical situations, enhancing the proactive monitoring and care provided to patients.

3. Cloud-based Storage and Analytics: Implementing cloud-based storage and analytics would allow for large-scale data storage, processing, and advanced analytics. This could include data aggregation, trend analysis, and predictive modeling, facilitating personalized healthcare recommendations and improved decision-making.

4. Mobile Application Development: Developing a dedicated mobile application would provide greater accessibility and convenience for healthcare providers and patients. The mobile app can allow real-time access to pulse data, personalized alerts, and interactive features for patient-doctor communication.

5. Integration with Electronic Health Record (EHR) Systems: Integrating the remote pulse monitoring system with existing Electronic Health Record (EHR) systems would enable seamless data sharing and consolidation. This would provide a comprehensive patient health record, ensuring continuity of care and facilitating data-driven treatment decisions.

6. Security and Privacy Enhancements: As remote patient monitoring involves the transmission and storage of sensitive health data, it is crucial to focus on robust security measures. Future developments should emphasize encryption techniques, secure data transmission protocols, and compliance with privacy regulations to safeguard patient information.

7. Clinical Trials and Validation Studies: Conducting clinical trials and validation studies to evaluate the effectiveness, accuracy, and usability of the remote pulse monitoring system would be essential. These studies can involve collaboration with healthcare institutions, clinicians, and patients to validate the system's performance in real-world scenarios.

8. Integration with Telemedicine Platforms: Integration with telemedicine platforms would enable synchronous or asynchronous video consultations between healthcare providers and patients. This would facilitate remote patient monitoring alongside virtual medical consultations, enhancing the overall healthcare experience and accessibility.

9. Scalability and Deployment in Healthcare Facilities: Future developments should focus on scalability and ease of deployment in various healthcare settings, including hospitals, clinics, and home healthcare. Modular and scalable architecture would ensure flexibility and adaptability to different environments and healthcare requirements.

10. User Experience and User Interface Refinement: Continual improvements in the user experience and user interface design of the Express MVC website and associated mobile applications would enhance the usability and acceptance of the remote pulse monitoring system by healthcare providers and patients.

V. CONCLUSION

In this research paper, we presented a remote pulse monitoring system that utilizes the ESP32 microcontroller, an Express MVC website, and a web API to enable efficient and effective remote monitoring of pulse data. The developed system successfully addressed the limitations of existing systems by providing real-time data visualization, seamless data transmission, and instant notifications to healthcare professionals.

Through the implementation of the ESP32, pulse data was collected accurately and transmitted to the web API using HTTP requests. The web API processed the data, generated appropriate responses, and facilitated seamless communication between the ESP32 and the Express MVC website. The Express MVC website, with its user-friendly interface, allowed healthcare providers to visualize live pulse data and interact with the system.

Additionally, the system incorporated a WhatsApp bot integration feature that triggered instant notifications to healthcare professionals based on specific response codes from the web API. This ensured timely intervention and improved patient care.

The successful integration and testing of the system demonstrated its feasibility and potential in remote patient monitoring. Healthcare providers can now remotely monitor patients' pulse data, visualize it in real-time, and receive timely notifications for critical situations. This has significant implications in terms of enhancing healthcare delivery, enabling early detection of abnormalities, and reducing the need for physical patient presence in healthcare facilities.

The results obtained from the experiments are discussed, emphasizing the strengths and limitations of the chosen models. The interpretation of the feature importance analysis is provided to gain insights into the risk factors associated with stroke occurrence. The potential clinical implications of the developed predictive model are explored, highlighting the importance of early detection and intervention in stroke prevention. The conclusion summarizes the findings and underscores the potential of machine learning techniques in stroke prediction, suggesting further research directions and improvements.

While this research paper presented a comprehensive solution, there are still avenues for future enhancements. The system can be further expanded to include additional vital signs monitoring, such as blood pressure or oxygen saturation levels. Furthermore, the integration with other communication channels or platforms can be explored to cater to different preferences and improve accessibility for healthcare professionals.

X . ACKNOWLEDGMENT

Thanks “VIT” for your incredible platform to publish papers.

REFERENCES

[1] Patel, K. R., & Patel, N. K. (2017). Remote patient monitoring system using IoT. International Journal of Engineering Research and Modern Education (IJERME), 2(2), 82-87.

[2] Nair, S., Kumar, M., & Gupta, S. (2020). IoT-based remote patient monitoring system for critical care. Journal of Medical Systems, 44(7), 129.

[3] Joshi, A., Rokade, K., Nemade, S., & Patil, P. (2018). IoT-based real-time health monitoring system using Arduino and ThingSpeak. In 2018 International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI) (pp. 640-643). IEEE.

[4] Verma, S., & Mahajan, K. (2018). A review on IoT-based remote health monitoring systems. In 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN) (pp. 375-380). IEEE.

[5] Devi, S. V., Reddy, P. P., Kumar, K. V., & Reddy, M. B. (2018). A cloud-based IoT healthcare system for monitoring heart rate and body temperature. Journal of Ambient Intelligence and Humanized Computing, 9(6), 1999-2012.