SMART HEALTH MONITORING SYSTEM

Minor Project report submitted to

Indian Institute of Information Technology Nagpur,

in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

with specialization of Internet of Things

by

Navya Srivastava (BT22ECI025) **Shashank Rao (BT22ECI031)**

Utkarsh Ghore (BT22ECI032) **Sahil Kumar (BT22ECI043)**

Under the guidance of

Dr. Mayur Parate

Indian Institute of Information Technology,

Nagpur* 441108 (*India)

2025

Department of Electronics and Communication Engineering

Indian Institute of Information Technology, Nagpur

Declaration-I

We, Navya Srivastava, Shashank Rao, Utkasrsh Ghore and Sahil Kumar, hereby declare that this project work titled **" Smart Health Monitoring System"** is carried out by us in the Department of Electronics and Communication Engineering of Indian Institute of Information Technology, Nagpur. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution/ University.

Date : 20th MAY 2025

Sr. No.	Enrollment No.	Names	Signature
1	**BT22ECI025**	**Navya Srivastava**	
2	**BT22ECI031**	**Shashank Rao**	
3	**BT22ECI032**	**Utkarsh Ghore**	
4	**BT22ECI043**	**Sahil Kumar**	

Declaration-II

We, Navya Srivastava (BT22ECI025), Shanshank Rao (BT22ECI31), Utkarsh Ghore (BT22ECI032) and Sahil Kumar (BT22ECI043), understand that plagiarism as any one or the combination of the following:

1. Uncredited verbatim copying of individual sentences, paragraphs or illustrations (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Uncredited improper paraphrasing of pages or paragraphs (changing a few words or phrases, or rearranging the original sentence order).
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did or wrote what. we have made sure that all the ideas, expressions, graphs, diagrams etc. that are not a result of my own work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.

We affirm that no portion of our work can be considered as plagiarism and we take full responsibility if such complaints occurs. We understand fully well the guide of the thesis may not be in a position to check for possibility of such incidences of plagiarism in this body of work.

Date : 20TH MAY 2025

Sr. No.	Enrollment No.	Names	Signature
1	**BT22ECI025**	**Navya Srivastava**	
2	**BT22ECI031**	**Shashank Rao**	
3	**BT22ECI032**	**Utkarsh Ghore**	
4	**BT22ECI043**	**Sahil Kumar**	

Electronics and Communication Engineering

IIIT, NAGPUR

Certificate

This is to certify that the project titled **"Smart Home Monitoring System",** submitted by Navya Srivastava, Shashank Rao, Utkarsh Ghore and Sahil Kumar in partial fulfillment of the requirements for the award of **Bachelor of Technology in Electronics and Communication Engineering with specialization of Internet of Things,** IIIT Nagpur. The work is comprehensive, complete and fit for final evaluation.

Date : 20TH MAY 2025

Dr. Mayur Parate

Assistant Professor, ECE Department, IIIT Nagpur

Dr. Harsh Goud

Head of Department, IIIT Nagpur

Acknowledgement

We take this opportunity to express our sincere thanks to our supervisor, Dr. Mayur Parate, for his unwavering guidance and encouragement throughout the project. We also thank the Department of Electronics and Communication Engineering at IIIT Nagpur faculty and staff for their support.

Abstract

The growing need for real-time monitoring systems in health has necessitated the adoption of Internet of Things (IoT) and edge computing technologies. This project introduces a Smart Health Monitoring System that is based on algorithms and edge computing employing Raspberry Pi to monitor key health parameters like ECG, body temperature, SpO_2 levels, fall detection and intake of meals and medication through Near Field Communication (NFC) technology. The system is designed to show early warnings and data visualization via an easy-to-use dashboard, improving patient care and facilitating remote health monitoring.

List of Symbols, Abbreviations or Nomenclature

1. **ECG**: Electrocardiogram
2. **SpO_2**: Blood Oxygen Saturation
3. **NFC**: Near Field Communication
4. **IoT**: Internet of Things
5. **API**: Application Programming Interface
6. **GPIO**: General-Purpose Input/Output
7. **MQTT/I2C/SPI/UART**: Communication Protocols
8. **PN532**: NFC Module Model
9. **AD8232**: ECG Sensor Model
10. **LM35**: Temperature Sensor Model
11. **MAX30100**: SpO_2 Sensor Model
12. **MPU6050**: Accelerometer Model

Table of Contents

CHAPTER I : INTRODUCTION

The growing need for real-time health monitoring systems has prompted integrating of Internet of Things (IoT), and edge computing technologies. This project offers a Smart Health Monitoring System that utilizes edge computing using Raspberry Pi 4B to monitor key health parameters like ECG, body temperature, SpO_2 levels, fall detection, and meal and medication intake using the Near Field Communication (NFC) technology.

The system is designed to give real-time alerts and data either from live sensors or simulated CSV datasets. ESP32 acts as the peripheral controller, publishing data to MQTT topics which the Raspberry Pi subscribes to. Predictions are served through JavaScript and visualized on a dashboard, offering caregivers actionable insights with the minimal latency.

CHAPTER II : LITERATURE REVIEW

The integration of Raspberry Pi with IoT platforms for health monitoring has been explored across multiple research works. The following studies provide insights into existing solutions and the evolution of smart healthcare systems leveraging edge computing and communication technologies.

1. **Rajkumar et al. (2017)**
   Proposed a health monitoring system utilizing Raspberry Pi as a core processing unit, interfaced with basic health sensors. The system demonstrated effective real-time data acquisition and storage, emphasizing low-cost implementation for primary healthcare needs [1].
2. **Bsoul et al. (2019)**
   Developed a smart healthcare framework using Raspberry Pi and IoT. Their system allowed continuous monitoring of vital parameters and displayed the results on a user-friendly interface, showcasing the feasibility of home-based health tracking [2].
3. **Kodali et al. (2018)**
   Introduced a Raspberry Pi and IoT-based solution for health data collection and alert generation. The system was capable of sending alerts via the internet when abnormal readings were detected, thereby highlighting the role of IoT in emergency health response [3].
4. **Springer (2022)**
   Explored the role of NFC technology in smart medicine tracking. The study presented methods to use NFC tags for recording medication adherence and integrating this data into mobile health applications, providing valuable insights for patient compliance monitoring [4].
5. **Hennebelle et al. (2023)**
   Presented *HealthEdge*, a comprehensive framework combining IoT, edge computing, and cloud technologies for predicting type 2 diabetes. Their work emphasized the advantages of on-device inference for real-time decision-making and reduced cloud dependency [5].

CHAPTER III : SYSTEM DESIGN AND IMPLEMENTATION

3.1 Hardware Components

1. **Raspberry Pi 4 Model B :** Serves as the edge server for the local processing.
2. **ECG Sensor (AD8232) :** Records electrocardiogram signals.
3. **Temperature Sensor (MAX30205) :** Measures temperature.
4. **Pulse Oximeter (MAX30102) :** Records blood oxygen saturation levels.
5. **Accelerometer (MPU6050) :** Detects falls based on motion analysis.
6. **NFC Reader (PN532 Module) :** Detects NFC tags related to meals and medications.
7. **NEO-6M GPS Module :** To detect the location of the person.

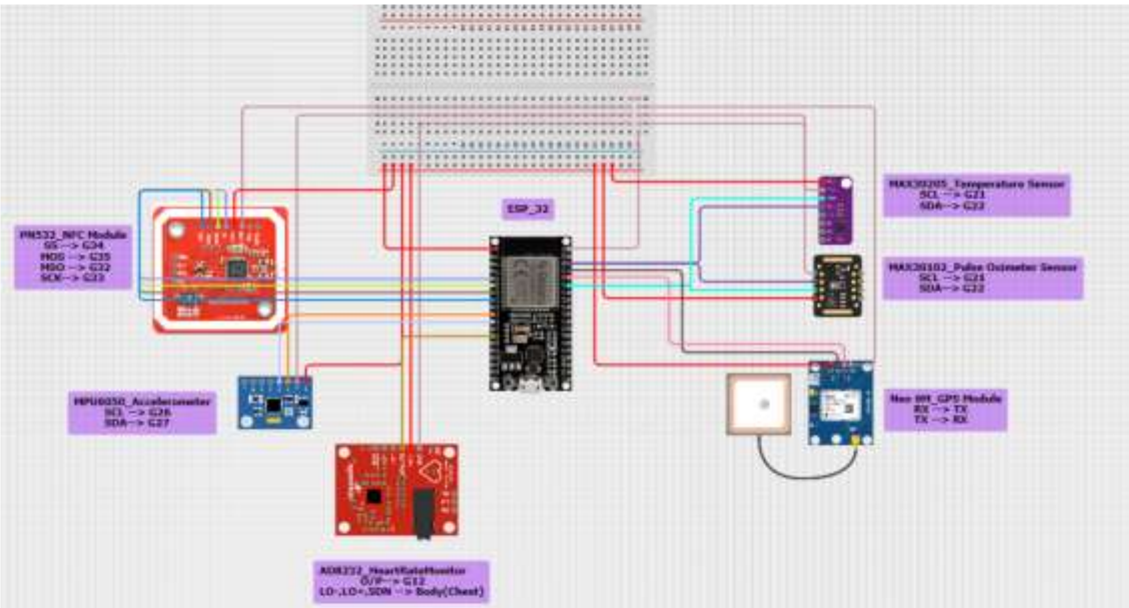

FIGURE 3.1 : Hardware Circuit Design for Smart Health Monitoring System Using ESP32

3.2 Data Flow

The Smart Health Monitoring System follows a layered architectural approach designed to ensure efficient data acquisition, processing, and storage. The architecture comprises three main tiers: an ESP32-based module for sensor data collection, a Raspberry Pi serving as an edge processing unit, and AWS DynamoDB as the cloud storage backend. A web application (under development) will later interact with the AWS database for real-time monitoring and analysis.

3.2.1. ESP32: Data Collection and Simulation

The ESP32 microcontroller simulates the functioning of a real-time wearable health monitoring device. It reads sensor data from pre-defined CSV files stored in the onboard LittleFS filesystem. These files emulate sensor behavior for different health parameters:

a) **High-Frequency Monitoring**:
   a. Accelerometer (MPU6050) and vitals (Heart Rate and SpO_2) are read every second, providing data essential for fall detection and continuous monitoring.

b) **Medium-Frequency Monitoring**:
   a. Body temperature data is updated every 15 seconds to balance accuracy with system efficiency.

c) **Low-Frequency/Event-Driven Monitoring**:
   a. GPS data is updated once per minute.
   b. NFC scans (for meals and medication intake) are simulated manually.
   c. ECG data is read periodically (~75 seconds) to simulate on-demand heart health checks.

Each data entry is encapsulated in a JSON format with a timestamp (ts_esp) and published to topic-specific MQTT channels (e.g., patient/PatientSim001/temperature) hosted on the Raspberry Pi.

3.2.2. Raspberry Pi: Edge Processing and Intelligence

Acting as the bridge between raw sensor data and structured cloud storage, the Raspberry Pi performs the following key roles:

a) **MQTT Broker and Listener**:
   a. Mosquitto is configured on the Pi to receive MQTT messages.
   b. A custom Python script (mqtt_processor.py) subscribes to all sensor topics and handles incoming data.

b) **Local Processing and Decision-Making**:
   a. **Fall Detection**: Real-time analysis of accelerometer and gyroscope data is conducted using threshold logic to identify potential falls.
   b. **Vital Sign Monitoring**: Temperature and SpO_2 data are checked against medical thresholds. Abnormal readings are flagged immediately.
   c. **Data Aggregation**: Vitals such as heart rate and SpO_2 are collected over a one-minute window and used to compute average, minimum, and series data.

c) **Alert Management**:
   a. Based on the analysis, alerts like "Fall Detected" or "High Temperature" are generated with timestamps.

d) **Cloud-Ready Data Preparation**:
   a. Data from different streams is consolidated into structured formats suitable for insertion into AWS DynamoDB tables.
   b. The Pi determines what to upload and when, prioritizing critical events and uploading periodic summaries every minute.

This approach minimizes bandwidth usage and ensures that only essential, processed data is stored in the cloud.

3.2.3. AWS DynamoDB: Cloud-Based Data Storage

Data from the Raspberry Pi is pushed into three DynamoDB tables with defined schemas:

a) **HealthData Table**:
   a. Stores 1-minute aggregated health metrics including temperature, heart rate, SpO_2, GPS location, and alerts.

    b. Keys: PatientID and timestamp.

b) **ECGData Table**:

    a. Logs ECG waveform data captured on demand.
    b. Includes raw ECG values and timestamps.

c) **NFCData Table**:

    a. Records all NFC-triggered events such as meal or medication intake.
    b. Contains relevant metadata like tag IDs and original scan timestamps.

This structured approach facilitates both detailed analysis and efficient querying for visual display and longitudinal tracking.

3.2.4 Admin Dashboard

For internal monitoring or debugging, a lightweight admin dashboard can be built where the JavaScript frontend interacts directly with AWS services:

a) The frontend uses AWS SDK for JavaScript or AWS Amplify to securely connect with:

    a. AWS API Gateway, which routes requests to
    b. AWS Lambda, where data is fetched from DynamoDB.

b) The retrieved data is dynamically rendered on the dashboard for inspection.

c) Visualization can be done using JavaScript libraries.

CHAPTER IV : Results and Discussion

4.1 System Functionality Overview

The Smart Health Monitoring System was successfully deployed and tested with the simulation of real-time data (datasets instead of physical sensors for now) and integrated into functional Flask API and Streamlit dashboard. The system showed correct prediction and status updates for multiplt health parameters and external inputs.

4.2 Backend Data Visualization via AWS Console

To validate the correct functioning of the Smart Health Monitoring System, backend data captured through MQTT and processed by the JavaScript was stored and visualized using AWS. Screenshots of key databases tables are presented below to demonstrate how patient information, sensor readings and event logs were systematically recorded.

Figure 4.1 : ECG Data Table to log ECG signals for each patient, with timestamps.

Figure 4.2 : Health Data Table to display heart rate, SpO2, fall status and alert messages linked to recent events.

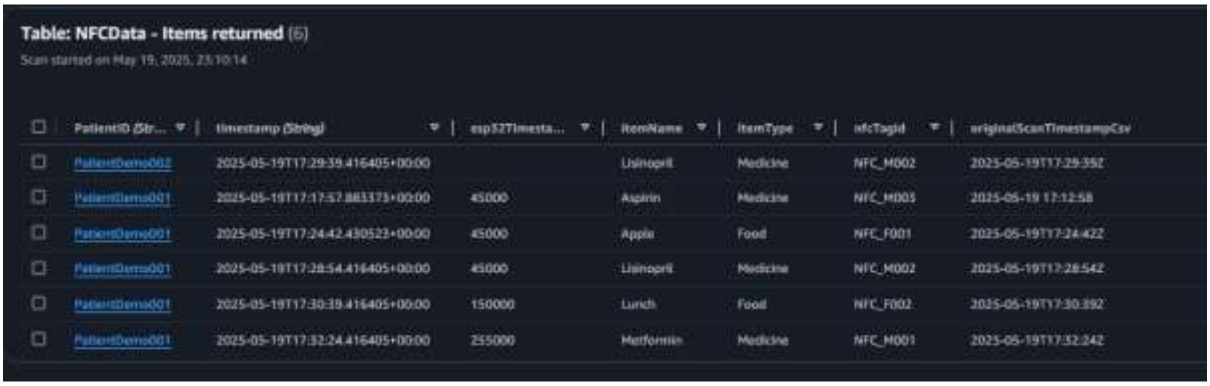

Table: NFCData - Items returned (6)
Scan started on May 19, 2025, 23:10:14

PatientID (Str...	timestamp (String)	esp32Timesta...	itemName	itemType	nfcTagId	originalScanTimestampCsv
PatientDemo002	2025-05-19T17:29:39.416405+00:00		Lisinopril	Medicine	NFC_M002	2025-05-19T17:29:39Z
PatientDemo001	2025-05-19T17:17:57.883373+00:00	45000	Aspirin	Medicine	NFC_M005	2025-05-19 17:12:58
PatientDemo001	2025-05-19T17:24:42.430523+00:00	45000	Apple	Food	NFC_F001	2025-05-19T17:24:42Z
PatientDemo001	2025-05-19T17:28:54.416405+00:00	45000	Lisinopril	Medicine	NFC_M002	2025-05-19T17:28:54Z
PatientDemo001	2025-05-19T17:30:39.416405+00:00	150000	Lunch	Food	NFC_F002	2025-05-19T17:30:39Z
PatientDemo001	2025-05-19T17:32:24.416405+00:00	255000	Metformin	Medicine	NFC_M001	2025-05-19T17:32:24Z

Figure 4.3 : NFC Data Table to capture the logs of meals and medication intake using NFC tags scanned by the patient.

Table: HealthData - Items returned (7)

Figure 4.4 : Health Data Table to link the various vitals and the GPS location of the patient.

4.3 Backend and Edge Integration

1. JavaScript hosts all trained models and sensor simulations.
2. Raspberry Pi functions as edge server.
3. All predictions computed on-device and served to dashboard.

4.4 Usability of Dashboard

1. Developed using Streamlit.
2. Updates visually every 5-10 seconds.
3. Shows ECG signal classification, Fall detection, SpO2, Temperature, GPS, NFC logs.

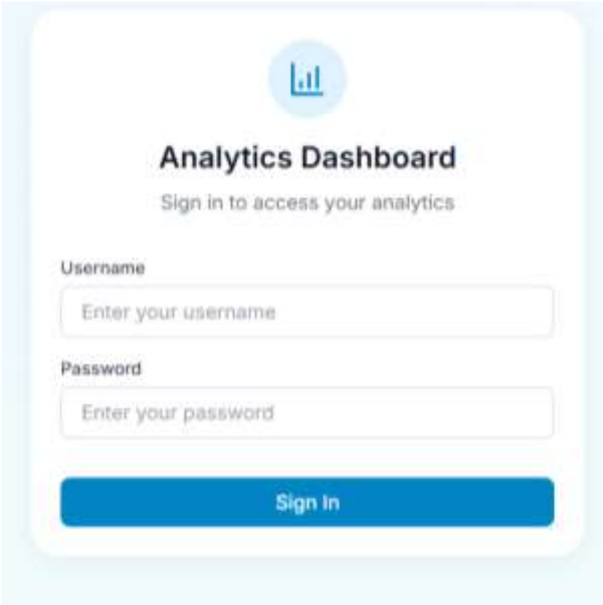

Figure 4.5 : Login page for the Smart Health Monitoring System's Analytics Dashboard.

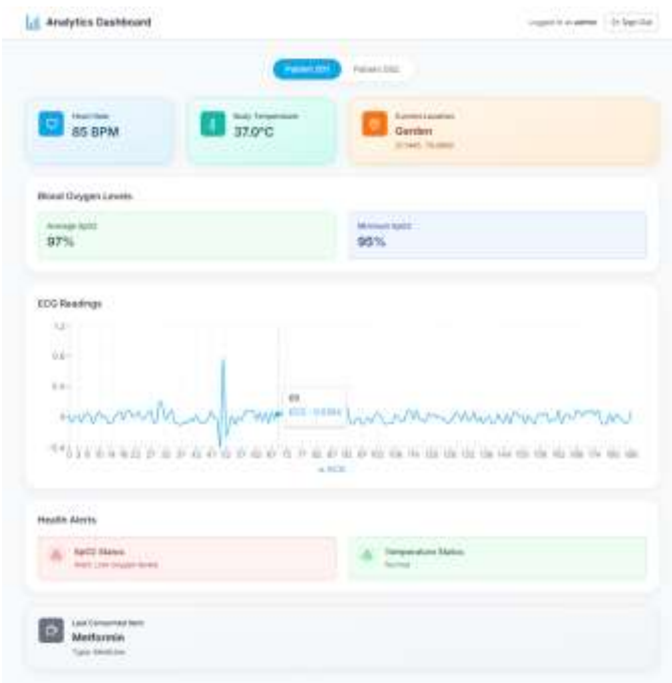

Figure 4.6 : Dashboard view for Patient 001 showing real-time vitals like heart rate, body temperature, location, SpO_2, ECG reading, health alerts, and medicine logs.

CHAPTER V : Conclusion and Future Work

5.1 Summary of the Work

The objective of this project was to create an end-to-end Smart Health Monitoring System based on machine learning and edge computing concepts. Meal and medication logging using NFC has been incorporated for patient health tracking completeness.

Major achievements:

1. Threshold-based temperature and SpO_2 alert logic.
2. Real-time dashboard in Streamlit for display of all important metrics.
3. Edge server simulation using Raspberry Pi-local setup.
4. NFC logging of daily patient activities such as food and medicine.

5.2 Conclusion

The system was able to successfully:

1. Apply local edge server concepts (Raspberry Pi simulated) for efficient computation.
2. Ensured extensibility: extensible to live sensors with replacement of CSV data stream.
3. Supported real-time monitoring and alert generation with minimum latency.
4. Offered user-friendly and auto-updating web dashboard for visualization.

5.3 Future Scope

1. Replace CSV simulation with live sensor data via GPIO.
2. Hosted on physical Raspberry Pi with cloud sync.
3. Add SMS/email alerts.
4. Integrate authentication and data encryption.
5. Expand to wearable or IoT-enabled prototype.

References

[1] Rajkumar, S., M. Srikanth, and N. Ramasubramanian, "Health monitoring system using Raspberry Pi," *Proceedings of the 2017 International Conference on Big Data, IoT and Data Science (BID)*, pp. 1–6, Chennai, India, Dec. 2017.

[2] Bsoul, M., et al., "Smart healthcare monitoring system using Raspberry Pi on IoT platform," *International Journal of Engineering and Technology (IJET)*, vol. 7, no. 2, pp. 234–238, 2019.

[3] Kodali, R. K., S. Soratkal, and P. Raj, "Health monitoring system using Raspberry Pi and IoT," *Proceedings of the 2nd International Conference on Inventive Systems and Control (ICISC)*, pp. 1–4, Coimbatore, India, Jan. 2018.

[4] Springer, "NFC Technology in Smart Medicine Tracking," *Lecture Notes in Computer Science*, Springer, 2022.

[5] Hennebelle, A., H. Materwala, and L. Ismail, "HealthEdge: A machine learning-based smart healthcare framework for prediction of type 2 diabetes in an integrated IoT, edge, and cloud computing system," *Sensors*, vol. 23, no. 4, pp. 1–20, 2023.

Appendices

Appendix A : Component Lists

S. No.	Component	Function
1	Raspberry Pi 4B	Edge device for data processing
2	ESP32	Sensor controller and MQTT publisher
3	MAX30205	Temperature sensing
4	MAX30102	Heart Rate and SpO_2 sensing
5	MPU6050	Fall detection using acceleration data
6	AD8232	ECG signal acquisition
7	PN532 NFC Reader	Detects meal/medicine intake via NFC
8	Neo-6M GPS Module	Location tracking
9	Jumper Wires & Breadboard	Interconnection and prototyping

Appendix B : Data Simulation Details

Parameter	File Name	Frequency	Samples in 5 Minutes
Temperature	temp_data.csv	Every 15 seconds	20
Accelerometer	acc_data.csv	Every 1 second	300
GPS	gps_data.csv	Every 1 minute	5
Heart Rate/SpO_2	hr_spo2.csv	Every 1 second	300 each
ECG	ecg10.csv	On-demand	2–3 sessions
NFC	nfc.csv	Manual trigger	Varies

Appendix C : Tools and Libraries Used

1. . **ESP32 (Arduino Environment)**

- **WiFi.h**: Network connectivity.
- **PubSubClient.h**: MQTT communication.
- **Optional sensor libraries**: Adafruit PN532, MPU6050, MAX3010x, TinyGPS++ for real sensor integration.

2. Raspberry Pi (Python Environment)

a) **paho-mqtt**: For subscribing to MQTT topics.
b) **boto3**: AWS SDK for inserting records into DynamoDB.
c) **numpy**: Used in calculations like averages and thresholds.
d) **json**, **datetime**: Handling message formats and timestamps.

3. Dashboard

a) **streamlit**: Fast and interactive UI for dashboards.
b) **boto3**: To retrieve data from AWS DynamoDB.
c) **pandas**: For structuring tabular data.
d) **numpy**: For additional calculations.
e) **plotly** or **matplotlib**: For generating dynamic visualizations.