Smart Patient Health Monitoring System

Final Report

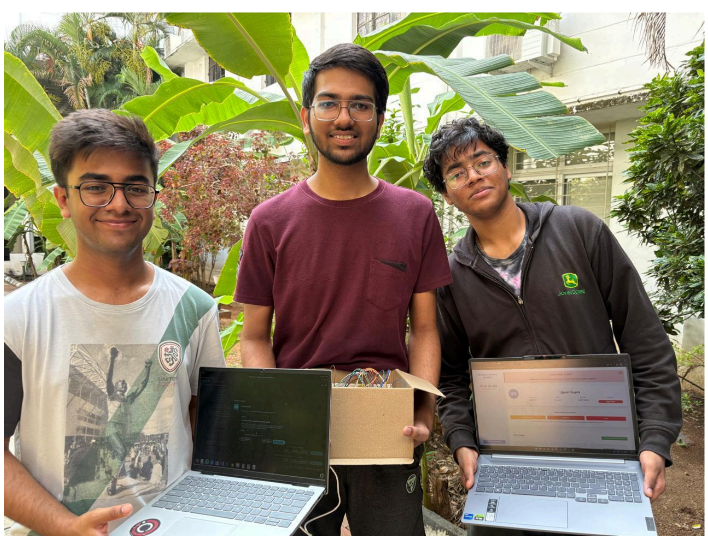
Team-62: Vansh Goyal (2024101044), Samarth Singla (2024101020), Ujjwal Gupta (2024117015)

Problem Statement

Post-surgical patients often require regular monitoring of vital signs, but not all cases demand continuous observation in an ICU setting. Keeping such patients admitted increases healthcare costs and strains hospital resources. There is a need for a smart, home-based monitoring system that can remotely track vital parameters, detect any abnormal changes in real-time, and alert caregivers or medical professionals instantly—ensuring patient safety without requiring prolonged hospital stays.

Proposed Solution

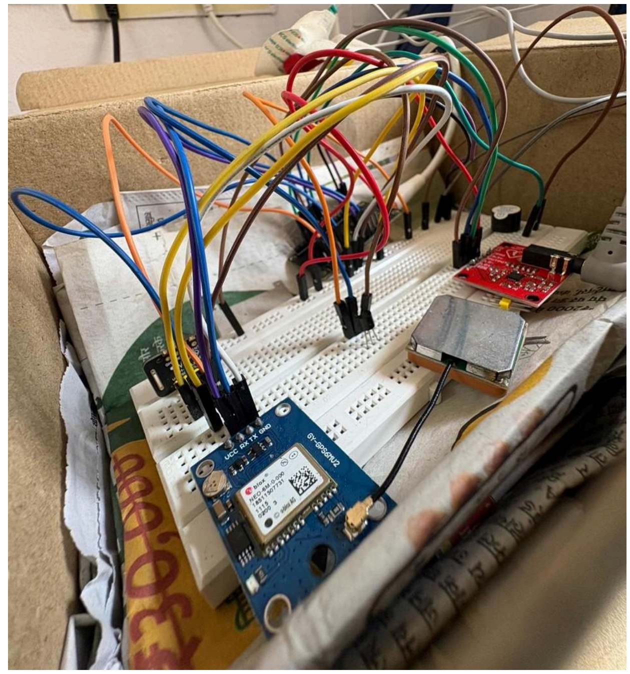
We propose a smart patient monitoring system that allows doctors to remotely track real-time vitals through a custom dashboard, which also generates alerts if a patient's condition becomes critical. This system can be integrated with emergency services, such as an ambulance dispatch, to enable timely response. Patients are equipped with a wearable kit (vest/sleeve) embedded with an ESP32 microcontroller, vital sign sensors, and a buzzer as an actuator for local alerts. Additionally, a machine learning model processes the real-time data in combination with historical records to assess patient status and predict potential health risks. We also implement live location tracking such that in case of patient emergency, adequate help can be sent timely on the correct location.



System Overview:

Sensors Used:

- Pulse Oximeter MAX30102
- LM-35 Body Temperature Sensor
- ECG Module AD8232
- Neo-6M GPS Module



Actuators Used:

- Active Buzzer (For Alerts)

Integration Platform:

- Thingspeak Cloud Services (Used http protocol)
- REST APIs

Implementation:

IoT Implementation (Sensors and Software):

- All sensors were individually tested and calibrated using the calibration methodology provided in the “Github Repository” link provided and calibrated.
- The data collected was sent to Thingspeak every 15 seconds (due to free account limitation). Location coordinates were also sent to update real-time location of the patient.
- Applied linear regression on data fetched from Thingspeak with the help of a prebuilt machine learning model using the “sklearn” module in Python. It took into account behaviour of recent past valid vitals (uptill last 100 data points).
- An alert status is sent to Thingspeak which is again fetched by the device to activate the buzzer accordingly.
- Due to restriction of uploading data on Thingspeak, devised a strategy to pause sensor data uploading code every minute so that correct alert status could be generated and uploaded by the machine learning model. Hence, every minute, we expect 2-3 data points uploaded on thingspeak and 1 alert generation.

Website Implementation:

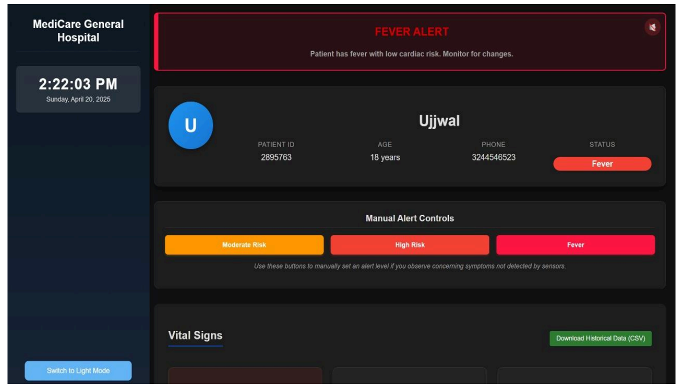
- We created a web dashboard providing real-time patient vitals.

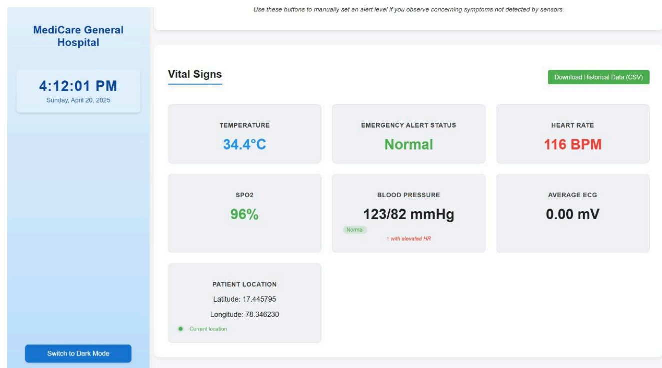
Backend:

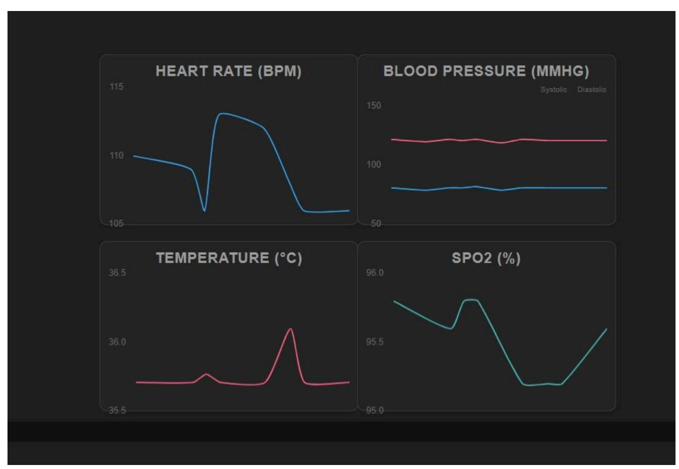
- Used MongoDB to store Doctor ID and Patient ID under that particular doctor. The Patient ID is actually the Thingspeak API Key used to fetch data of that particular patient from a particular channel.
- After the patient ID is fetched successfully from the database it is used to fetch the patient's vitals from the thingspeak channel.
- Used FastAPI to handle all data fetch and upload requests.

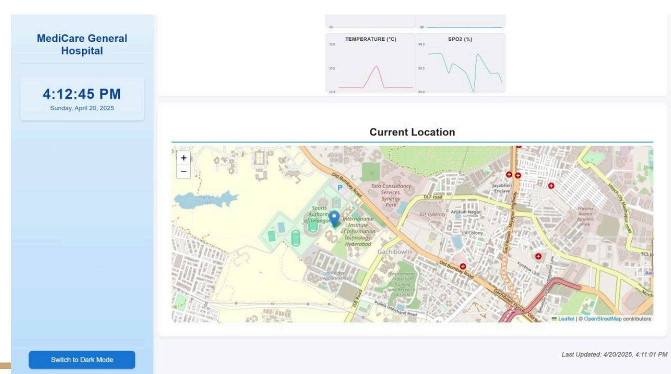
Frontend:

- Our web dashboard has an integrated alert system for both fever and heart attack stages and a privilege to set manual alerts if the Doctor feels any abnormality.
- Medication Alert is available for patients in case of some emergency. The Doctor can add/remove medicines from the list at any time.
- Graph representation of each data field as well as option to download previous records of patients by just one click.



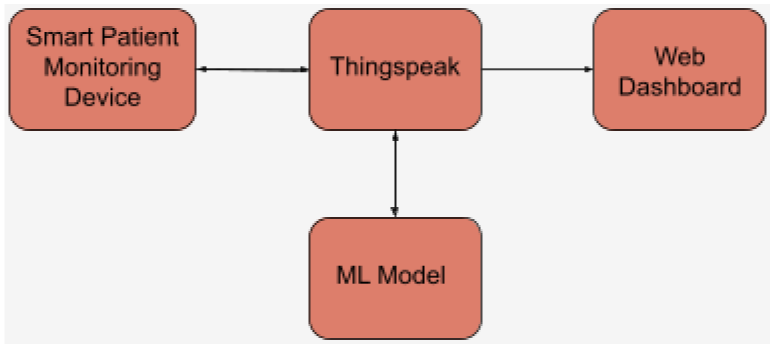




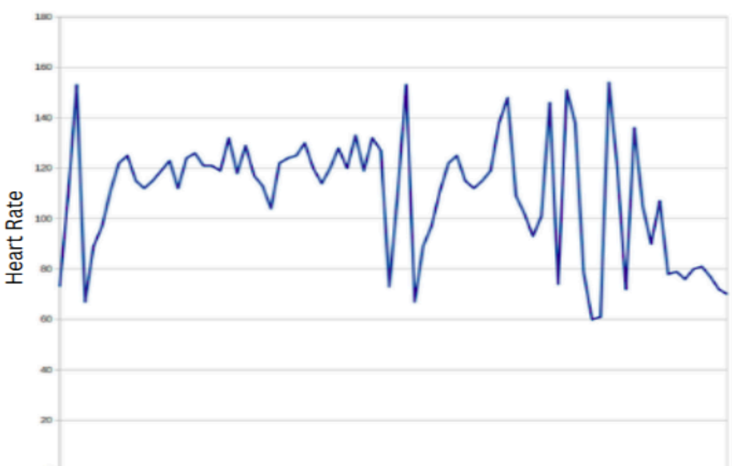


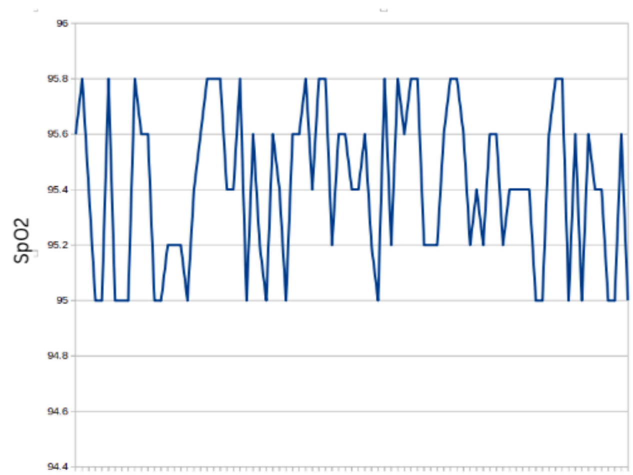
- Real time location updates of the patient to send help as soon as possible in case of emergency.

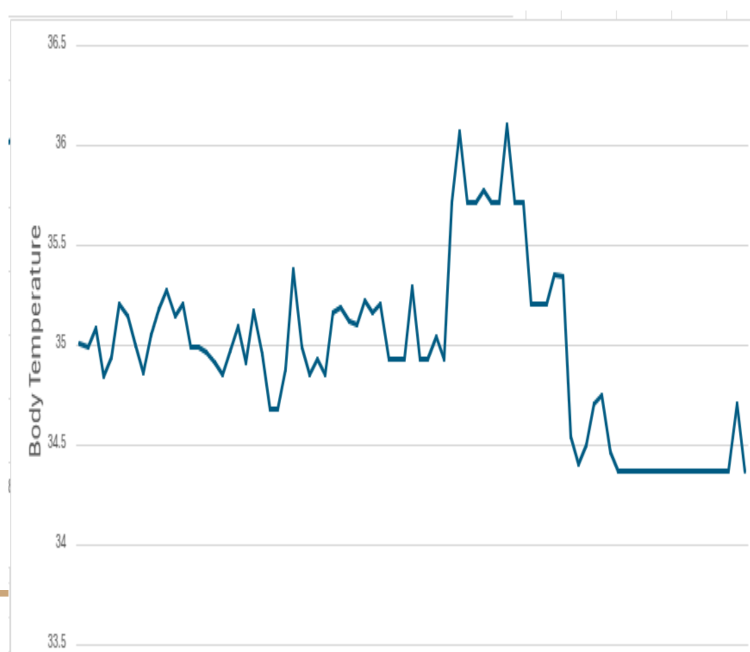
System Architecture:

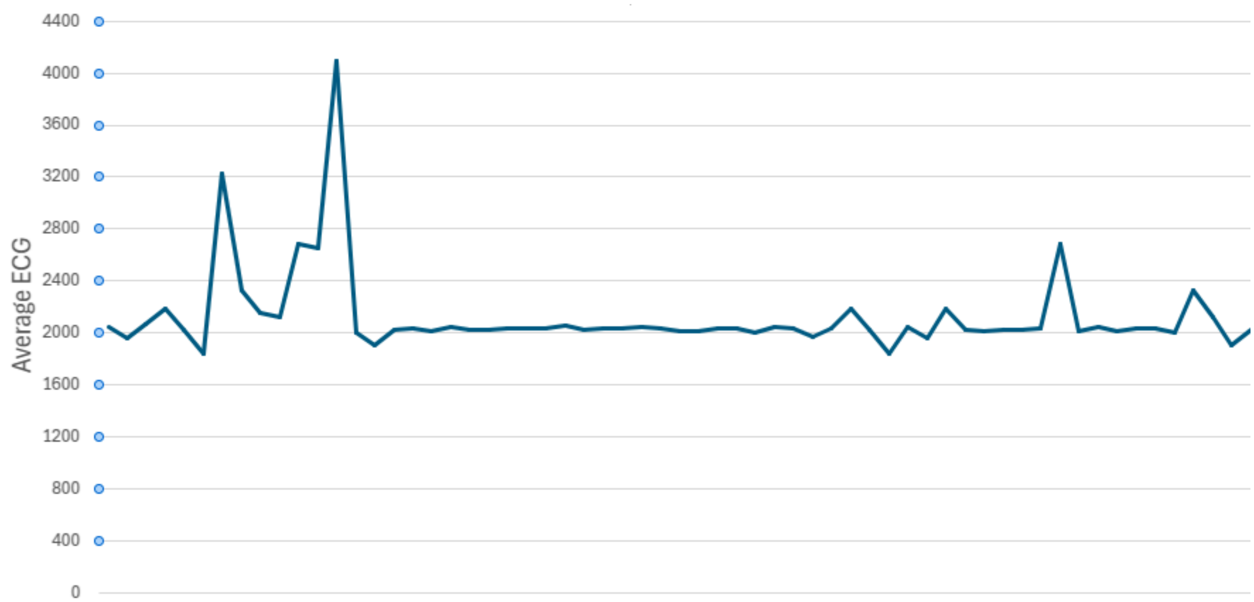


Data Analysis:









Challenges Faced:

Major challenges faced during the project were as follows:

1. **Configuring and Calibrating Sensors**
2. **Library Issues:** Few changes had to be made to the library of some sensors to use them. Changes included verifying function names and what they do and changing some functions from private to public class.
3. **Connection Issues:** There were connection and voltage issues. LM35 was not being able to connect properly due to which voltage continued to vary around it giving false values sometimes even after calibration. Soldering issues were also there.
4. **Neo-6M GPS Module Configuration:** Its configuration required us to get the device out in the open sky so that it can catch signals from the satellite. (However it did get good signal once while under the roof).
5. **Improper Uploading of Data due to Thingspeak restriction:** Had to devise a strategy of pausing the program once every minute to allow proper alert status uploading.
6. **Other hardware errors:** Improper boot mode, fatal error during booting, etc.
7. **Backend not responding:** We used MongoDB and fast api to store doctors' and fetch patient's data from Thingspeak. We were able to fetch records from the database but it was not getting displayed on the website because we missed to integrate some important Thingspeak services.

8. **Running Alert System:** Alert System includes a processing model which fetches records and runs algorithms to find risk status of the patient. Integrating it with the project was a difficult task.

Scalability:

We can scale multiple features of our system with better resources:

1. **Multiple Users:** We can modify the source code such that when the device boots for the first time, it automatically creates a Thingspeak channel (i.e. creates a new user in the database). The device will be uniquely identified exploiting manufacturing properties of ESP32.
2. **Designing a compact PCB:** This can help in making the device wearable. Since it would be easy to fit a pcb in some kind of special sleeve, we would have a complete wearable device that is now ready to function in the real world.
3. **Using Thingspeak Paid Account:** This helps uploading of data quickly and more efficiently.
4. **Hosting our own web server:** This we think would be a better option since we will be able to control the flow of our system more efficiently adding asynchronous functions.
5. **Mobile Application for Patient:** This should be done to alert the patient on his mobile phone to inform if anything is wrong and advice on what he/she should do as preventive measures. A whole lot of new features can be added if we make a mobile application.
6. **Additional Sensors:** We can use gyroscopes for fall detection and some skin conductance sensors to measure dehydration levels in the patient's body.

Lessons Learnt:

We learnt a lot from both IoT and Software parts while working on this project.

1. The project really helped us to put our thinking caps on and make something that taught us a lot technically, mentally and socially.
2. We got to know about what happens when a heart attack is about to happen. Got to know the principal ECG works upon.
3. We understood the functionality of the sensors used in the patient monitoring system.
    - How to process and calibrate any sensor to get accurate readings?
    - How can we communicate data to various software systems in real-time?
    - How to handle various kinds of errors (connection, configuration, network)?

4. We learnt about how to build a proper website with backend and frontend which can fetch and process data from MongoDB and Thingspeak whenever needed. We also learnt the use of React-like frameworks , http protocol and FAST APIs.
5. Working on a single project in a team of three taught us teamwork that will help us working on group projects in future for a company or startup.

Useful Links:

1. **GitHub Repository:** https://github.com/TECHIE-TITAN/Smart-Patient-Health-Monitoring-System
2. **Images and Videos:** IoT Project Videos and Images