Chapter 1

INTRODUCTION

In this project an IOT based patient health monitoring system is designed using MAX30100, LM35 and MPU6050 sensors. MAX30100 sensor is designed to measure Heart Rate; the LM35 sensor will provide body temperature measurements and MPU6050 will provide physical movement and orientation such as fall detection or motion analysis.

1.1Background

In this project an IOT based patient health monitoring system is designed using MAX30100, LM35 and MPU6050 sensors. MAX30100 sensor is designed to measure Heart Rate; the LM35 sensor will provide body temperature measurements and MPU6050 will provide physical movement and orientation such as fall detection or motion analysis. These sensors are connected to microcontroller (NodeMCU) and the data found is viewed in the serial monitor, and also pushed to Firebase at the same time. This Web-based solution allows health care providers to remotely view and track patient health data via a real-time dashboard. This technology can have a great impact on the provision of health care, particularly on in underserved or rural areas where medical facilities are scarce. Therefore, the developed system provides a low-cost, scalable, and robust solution for current healthcare monitoring requirements.

Intelligent health monitoring systems utilize implanted sensors, microcontrollers, and the cloud framework to monitor and transmit patient information in real time Monitoring vital signs including heart rate, body temperature, blood oxygen level (SpO_2), and physical activity continuously with these systems can have a significant impact. Advances in Internets of Things (IoTs) have led to more sophisticated and scalable health monitoring solutions available remotely.

These systems prevent premature hospital admissions and are capable of predicting early stages of health deterioration leading to early medical intervention, ultimately minimizing healthcare expenditure. In conventional approaches, there are platforms (such as ThingSpeak) that provide the storage of data and isolation. But today we have more flexible real-time solutions like Firebase which are becoming more and more popular because they are very scalable and much easier to setup with mobile.

The rise of Internet of Things (IoT) technology has changed the way we monitor and care for patients. With improved internet access and smarter devices, it is now possible to track a patient's health 24/7 using IoT-based health monitoring systems. These devices collect real-time data, monitor vital signs, and allow remote access to patient information. This not only helps in continuous monitoring but also supports quick emergency responses. Traditional patient monitoring in hospitals is often slow, requires manual checks by medical staff, and can lead to errors. Families must also visit hospitals to get updates, which can be stressful. Many hospitals still use paper records, which are prone to mistakes and missing data.

This wastes time and can harm patient care. An IoT-based health monitoring system offers a better solution by enabling remote monitoring, digital record keeping, and quick health updates. This study aims to improve healthcare services by using IoT to overcome current challenges. With features like real-time monitoring, smart data collection, and instant alerts, IoT systems are well-suited for modern healthcare. Even though earlier studies have shown great results, adoption has been slow. This work aims to build a more reliable, updated, and innovative healthcare system that meets today’s needs for saving lives and advancing medical technology.

This study contributes to the healthcare field by developing an Internet of Things (IoT)-based patient health monitoring system that allows remote tracking of vital signs and provides quick medical response during emergencies. The system uses heartbeat and temperature sensors connected to an NodeMCU, which send real-time data to an IoT web platform. This platform displays the patient's health status continuously. The proposed system offers a modern, efficient, and effective way to monitor patient health, helping to improve overall healthcare services and leading to better quality of life and health outcomes for society. IoT-based health monitoring systems offer a more flexible and cost-effective solution by leveraging smart sensors, wireless communication, and cloud platforms.

Traditional patient monitoring in hospitals is often slow, requires manual checks by medical staff, and can lead to errors. Families must also visit hospitals to get updates, which can be stressful. Many hospitals still use paper records, which are prone to mistakes and missing data. This work aims to build a more reliable, updated, and innovative healthcare system that meets today’s needs for saving lives and advancing medical technology.

1.2 Problem Statement

Traditional health monitoring methods often require frequent hospital visits, wired equipment, and manual supervision, which can be impractical for continuous observation, especially for elderly patients or individuals with chronic illnesses. With the advancement of Internet of Things (IoT) technology, there is a growing need for compact, low-cost, and real-time health monitoring systems that can track vital signs remotely and efficiently.

The ESP8266 NodeMCU microcontroller is widely used in IoT applications due to its built-in Wi-Fi and low power consumption. However, it faces limitations when multiple I2C-based sensors are connected simultaneously, resulting in unstable performance and insufficient power supply. In this system, three sensors are used: the LM35 for temperature sensing, the MAX30100 for heart rate and SpO2 monitoring, and the MPU6050 for fall detection. Connecting all three sensors to a single ESP8266 leads to issues such as voltage drops and communication conflicts, especially since both the MAX30100 and MPU6050 rely on the I2C protocol.

To address these challenges, the proposed system distributes sensor data to multiple cloud platforms: the MAX30100 sends real-time heart rate and oxygen data to the Blynk IoT platform for mobile-based monitoring, while the LM35 and MPU6050 transmit temperature and motion data to Firebase, which is then visualized on a web dashboard. This hybrid data routing approach reduces the communication load on the ESP8266 and ensures stable, real-time monitoring of patient health parameters.

This project aims to design and implement a reliable, low-power IoT-based health monitoring system using ESP8266 and multiple sensors while overcoming power and communication limitations inherent to the microcontroller. IoT-based system is in charge of providing knowledge from an environment to a non-expert user. IoT-based system can be used in different environments, so it needs to be able to address many heterogeneous devices. Thus, a major concern within developing an IoT-based system is how to handle the interaction with the heterogeneous devices for non-expert users. This concern can be addressed by a middleware layer between devices and non-expert users.

1.3 Objectives

The primary objective of this project is to develop and implement a reliable, intelligent health monitoring system using the ESP8266 NodeMCU microcontroller. This system is intended for continuous, real-time tracking of a patient's critical physiological parameters, including body temperature, heart rate, and fall detection. In this design, three sensors are deployed: the LM35 analog temperature sensor is used to monitor the body temperature of the patient; the MAX30100 optical sensor is employed to measure both heart rate and blood oxygen saturation (SpO_2); and the MPU6050 sensor is used to detect body motion and possible falls through its integrated accelerometer and gyroscope functionalities. These sensors collect data from the patient's body and transmit the information wirelessly through the ESP8266.

One of the major goals of the project is to overcome the limitations of power and data congestion associated with connecting multiple I^2C-based sensors to a single microcontroller. To address this, the system architecture separates the data flow: the MAX30100 sensor sends heart rate and SpO_2 data to the Blynk IoT platform, allowing real-time visualization on a mobile application, while the LM35 and MPU6050 sensors transmit their data to Firebase, where it is stored and fetched into a custom-built web dashboard. This hybrid platform approach ensures better performance, efficient power management, and minimal I^2C communication conflicts, while allowing multi-channel monitoring.

Additionally, the system is designed to store and analyze the patient's data in the cloud, providing historical trends and enabling doctors or caregivers to review vital signs over time. The data can be used for early diagnosis by analyzing patterns, detecting anomalies, and triggering alert notifications. These alerts help in taking timely medical action in case of abnormal readings, such as elevated body temperature or irregular heart rate. The system also aims to present health information in an easily understandable and user-friendly format to both patients and medical professionals through mobile and web interfaces. This contributes to enhanced patient engagement, remote health assessment, and preventive healthcare delivery.

The main objective of this project is to develop an intelligent and reliable patient health monitoring system using the ESP8266 microcontroller integrated with multiple biomedical sensors. The system is designed to continuously monitor vital physiological parameters such as body temperature, heart rate, and fall detection, providing real-time data collection and analysis. The LM35 sensor is used to measure body temperature, the MAX30100 sensor is used to monitor heart rate and blood oxygen saturation (SpO_2), and the MPU6050 sensor is employed to detect physical motion and sudden falls through its built-in accelerometer and gyroscope. The collected data is transmitted wirelessly to cloud platforms—Blynk for real-time monitoring of heart rate and SpO_2, and Firebase for long-term storage and dashboard visualization of temperature and motion data.

This configuration not only ensures real-time access to patient health information but also allows early detection of abnormalities through automated alerts. Furthermore, the system aims to present health-related information in an intuitive and user-friendly format, enabling both patients and healthcare providers to interpret and respond to the data effectively. By facilitating continuous health surveillance, the system supports timely intervention and preventive care, ultimately improving the quality of patient monitoring and reducing the burden on healthcare facilities.

The overarching objective of this project is to design, develop, and implement an intelligent IoT-based patient health monitoring system that enables continuous, real-time assessment of essential physiological parameters. The system utilizes the ESP8266 NodeMCU, a compact microcontroller with built-in Wi-Fi, as the central unit for data collection and wireless transmission. The key biomedical parameters monitored include body temperature, heart rate, blood oxygen saturation (SpO_2), and body motion or fall detection, which are critical indicators in assessing a patient's health status, especially for elderly individuals, patients in post-operative recovery, or those requiring chronic care.

To achieve these goals, the system employs a combination of three sensors: the LM35 temperature sensor to accurately measure the body's thermal condition, the MAX30100 pulse oximeter sensor to detect heart rate and blood oxygen levels using optical absorption techniques, and the MPU6050 sensor to capture dynamic motion data and recognize sudden changes in acceleration that may indicate falls or unusual physical activity. These sensors are carefully integrated into the system with optimized power distribution and I^2C communication management to ensure stable and reliable operation.

1.4 Scope of Study

The growing advancement in smart healthcare technology has led to the integration of Internet of Things (IoT) devices into health monitoring systems, enabling doctors and caregivers to track a patient’s physiological data remotely and in real time. This project focuses on the development and implementation of a smart health monitoring system using the ESP8266 microcontroller, which supports wireless communication and IoT functionality. The system incorporates three key biomedical sensors: the LM35 analog temperature sensor, the MAX30100 pulse oximeter and heart rate sensor, and the MPU6050 accelerometer and gyroscope. The primary aim is to monitor body temperature, detect falls, and measure heart rate and oxygen levels, thus ensuring timely alerts and responses in the event of abnormal conditions.

Due to hardware limitations and power constraints inherent in using multiple I^2C devices with a single ESP8266 module, the system is designed to split data transmission across platforms. The MAX30100 sensor, responsible for heart rate and SpO_2 monitoring, transmits data to the Blynk IoT platform, which provides real-time display and remote access via a mobile application. Meanwhile, the LM35 and MPU6050 sensors are configured to send their data to Google Firebase, a cloud-based database from which the information is retrieved and visualized in a custom dashboard. This architecture effectively avoids I^2C data conflicts and overload on the ESP8266’s limited power capabilities, while also enhancing reliability and communication efficiency.

This health monitoring system is designed to address pressing needs in modern healthcare, such as early disease detection, fall prevention, and remote patient supervision, especially for individuals suffering from chronic diseases like heart conditions, diabetes, or neurological disorders. By using cloud storage and real-time alerts, caregivers can be instantly informed when a patient's readings exceed predefined thresholds—such as elevated temperature, irregular heartbeats, or sudden body movements indicating a fall. Furthermore, the system significantly reduces the need for physical hospital visits, thereby lowering healthcare costs and improving the quality of care.

Mobile Health (mHealth) and Electronic Health (eHealth) technologies are rapidly transforming the healthcare landscape. With IoT-enabled sensors, vital signs can be collected, analyzed, and shared securely via cloud platforms using protocols such as Wi-Fi, Bluetooth Low Energy, or ZigBee. This system allows doctors and family members to monitor patients across distances, and receive alerts in life-threatening scenarios, thus facilitating real-time decision-making and intervention. Devices like these are instrumental in building a more proactive healthcare model, where continuous monitoring replaces periodic check-ups, enhancing the chances of early diagnosis and better outcomes. In the proposed system, the fusion of real-time data processing, remote cloud-based storage, and user-friendly interfaces exemplifies a cost-effective and scalable solution for smart healthcare delivery.

1.5 Significance of the Study

The integration of multiple vital health sensors with a single ESP8266 microcontroller, enabling real-time monitoring of essential physiological parameters such as body temperature, heart rate, blood oxygen level, and motion data. By utilizing the LM35 temperature sensor, MAX30100 pulse oximeter, and MPU6050 accelerometer/gyroscope, the system aims to provide a comprehensive health monitoring solution accessible remotely via IoT platforms like Blynk and Firebase.

The approach addresses practical challenges related to sensor data management and power supply optimization when multiple I2C sensors are connected to a single microcontroller. Splitting the data streams—sending MAX30100 data to the Blynk IoT platform and LM35 and MPU6050 data to Firebase—enhances system reliability and efficiency, allowing for flexible and scalable data visualization through dashboards.

Furthermore, the study highlights the importance of IoT in healthcare by enabling continuous, remote patient monitoring, which can facilitate early detection of health anomalies and support timely medical intervention. This contributes to improving patient care quality, reducing hospital visits, and promoting health awareness. Overall, this project demonstrates a practical and cost-effective method for integrating multi-sensor health monitoring with cloud platforms, potentially benefiting healthcare providers, patients, and researchers interested in IoT-based health systems. The development of a health monitoring system using the ESP8266 microcontroller integrated with multiple sensors—LM35 for temperature measurement, MAX30100 for pulse oximetry (heart rate and blood oxygen levels), and MPU6050 for motion tracking—offers significant contributions in both the technological and healthcare domains.

Technological Significance

This study demonstrates the practical integration of diverse physiological sensors in a compact and low-cost IoT platform. The ESP8266, with built-in Wi-Fi capabilities, serves as an effective node for transmitting sensor data wirelessly to cloud platforms such as Blynk and Firebase. Using the I2C communication protocol, multiple sensors are connected efficiently, though challenges such as power constraints and bus limitations are addressed by strategic data routing—MAX30100 data is sent to Blynk, while LM35 and MPU6050 data are transmitted to Firebase. This division helps optimize network traffic and power consumption, improving overall system stability and responsiveness.

The research provides insights into sensor fusion techniques, cloud data management, and real-time data visualization in dashboards, essential for building scalable IoT healthcare applications. Moreover, this system serves as a foundation for future enhancements, including integrating additional sensors, employing edge computing, and improving power management for extended battery life.

Practical and Healthcare Significance

From a healthcare perspective, the system enables continuous, non-invasive monitoring of vital signs critical to patient health assessment. Real-time monitoring of body temperature, blood oxygen saturation (SpO2), heart rate, and physical activity helps in early detection of abnormalities such as fever, hypoxia, arrhythmia, or falls. This is particularly valuable for elderly patients, individuals with chronic illnesses, or those recovering post-surgery.

Remote monitoring capabilities reduce the need for frequent hospital visits, lowering healthcare costs and minimizing exposure to infectious diseases—a factor especially relevant in pandemic scenarios. The availability of real-time data on accessible platforms empowers healthcare professionals to make timely, informed decisions and allows patients to engage actively in managing their health.By providing an affordable and scalable health monitoring solution, the project supports wider adoption of telemedicine and IoT-based healthcare services in resource-limited settings. This promotes equitable healthcare access and supports public health initiatives aimed at monitoring and controlling chronic disease

Chapter 2

LITERATURE REVIEW

The Internet of Things (IoT) is transforming the way safety and health monitoring systems are designed and deployed, offering real-time sensing, data sharing, and intelligent decision-making. In safety-critical applications, especially in healthcare, IoT allows for continuous monitoring of individuals without the need for constant human supervision**.**

2.1 Health Monitoring System Technologies

The health monitoring system is built using modern embedded and IoT technologies to track vital signs such as body temperature, heart rate, blood oxygen level, and physical movement. At the core of the system is the ESP8266 NodeMCU microcontroller, which is a low-cost, Wi-Fi-enabled board ideal for IoT-based applications. It acts as the central controller, interfacing with all the sensors and handling data transmission to cloud platforms.

Three sensors are integrated into the system: the LM35 temperature sensor, the MAX30100 pulse oximeter, and the MPU6050 accelerometer and gyroscope module. The LM35 is used to measure body temperature and provides an analog voltage output that is proportional to the temperature in Celsius. The MAX30100 measures both heart rate and blood oxygen saturation (SpO_2) using infrared and red LEDs along with a photodetector. The MPU6050 sensor is used to detect motion and orientation by providing data from a 3-axis accelerometer and 3-axis gyroscope. These sensors are connected to the ESP8266 through the I2C protocol, which enables communication using only two data lines (SDA and SCL). However, since all three sensors use I2C and draw power from the ESP8266, the system experiences power limitations when all are connected simultaneously.

To address this challenge, the system is designed to divide data transmission between two platforms. The MAX30100 sends its data to the Blynk IoT platform, which allows real-time heart rate and SpO_2 monitoring on a mobile app interface. Meanwhile, the LM35 and MPU6050 transmit their data to the Firebase Realtime Database. Firebase is a cloud-based NoSQL database that supports real-time data synchronization, allowing sensor data to be stored and displayed on a custom web.

2.2 Internet of Things (IoT) in Safety Monitoring

The Internet of Things (IoT) is transforming the way safety and health monitoring systems are designed and deployed, offering real-time sensing, data sharing, and intelligent decision-making. In safety-critical applications, especially in healthcare, IoT allows for continuous monitoring of individuals without the need for constant human supervision. The proposed health monitoring system using ESP8266 and multiple biomedical sensors is a practical application of IoT in this domain, where safety monitoring becomes intelligent, wireless, and easily accessible.

In this system, the ESP8266 NodeMCU serves as the core IoT device that connects three different sensors: LM35 for temperature measurement, MAX30100 for pulse rate and SpO_2 monitoring, and MPU6050 for motion tracking. These sensors continuously collect health-related data, which is then transmitted wirelessly to cloud platforms using Wi-Fi. By integrating Blynk and Firebase services, the project leverages IoT for not only data collection but also for real-time visualization, remote access, and safety alerts.

The MAX30100 sensor communicates with the Blynk IoT platform, allowing users to monitor cardiovascular conditions like heart rate and oxygen saturation through a mobile app. This is particularly useful for patients with respiratory or cardiac issues. At the same time, the LM35 and MPU6050 sensors send data to Firebase, where it is stored and later fetched for display on a custom-built dashboard. This dashboard enables healthcare providers, caregivers, or the users themselves to track trends in temperature and physical activity over time. This integration improves safety by allowing immediate response in case of irregular body temperatures or unexpected motion patterns, such as falls.

One of the key advantages of using IoT in safety monitoring is **mobility and remote accessibility**. Data from the sensors can be accessed from anywhere in the world, removing the limitations of physical presence and manual measurement. This is extremely beneficial in rural areas, old age homes, or for individuals under home quarantine who require constant monitoring without being in a hospital environment. Additionally, the use of Firebase allows for real-time data logging and historical record-keeping, which can be crucial for diagnosing long-term conditions.

Another important aspect is **automation and intelligent alerting**. With IoT, thresholds can be set for vital signs—such as temperature or heart rate—and notifications or alerts can be triggered when values go beyond safe limits. This can help in early detection of health deterioration and allow timely intervention, potentially preventing medical emergencies. For example, if the LM35 detects a sudden spike in temperature or if the MPU6050 detects a fall (sudden acceleration or abnormal angle), the system can alert caregivers immediately.

However, challenges such as power management and communication overload exist, especially when multiple I2C devices are connected to a single microcontroller like the ESP8266. Since all three sensors use I2C communication and require stable voltage, the system needs to be optimized. In this project, data is split between Blynk and Firebase to reduce communication load and ensure reliable operation. This design strategy is an example of how IoT-based safety systems must be intelligently architected to balance performance, power, and data accuracy.

2.3 Firebase and Real-Time Data Storage

Firebase, a powerful cloud-based platform developed by Google, plays a central role in enabling real-time data storage and synchronization in this health monitoring system. As a Backend-as-a-Service (BaaS), Firebase provides a scalable and reliable infrastructure that supports the collection, storage, and retrieval of data in real-time, making it ideal for healthcare-related IoT applications. In this project, Firebase is used to manage data collected from two key sensors: the LM35 temperature sensor and the MPU6050 accelerometer and gyroscope sensor. These sensors are responsible for capturing body temperature and physical movement, respectively, which are essential indicators for assessing an individual's health condition and physical safety.

The ESP8266 microcontroller reads the analog and digital outputs from these sensors and transmits the data to the Firebase Realtime Database over Wi-Fi. The database automatically updates in real time, allowing any connected application or dashboard to reflect the latest sensor readings instantly without requiring manual refreshes. This feature is crucial in a health monitoring context where timely updates can be lifesaving, such as detecting a fever spike, a fall, or abnormal motion patterns that may indicate distress or a medical emergency.Firebase stores the sensor data in a JSON tree format, which is lightweight and easy to navigate. This structure allows developers to categorize data efficiently based on attributes like user ID, sensor type, date, and time. For instance, data from the LM35 can be stored with a timestamp and associated with a particular patient profile, while movement data from the MPU6050 can be logged in a sequence for motion analysis.

This level of organization is highly beneficial for building a long-term, user-specific health tracking system and for generating meaningful health trends or reports. A key feature of Firebase is its real-time synchronization across devices. Whether accessed via a smartphone, tablet, or web dashboard, the Firebase database ensures that all users see the most current data instantly. This is especially important in remote health monitoring, where doctors or caregivers need to access a patient's vital data from different locations. Furthermore, Firebase supports push-based data communication, which means the system can notify users automatically when certain thresholds are crossed—for example, if a patient's temperature exceeds a preset safe limit, an alert can be triggered through the dashboard or via a separate notification system.

Security is another important advantage of Firebase. The platform supports user authentication and fine-grained access control rules, ensuring that only authorized users can view or modify health data. This is critical in applications that deal with sensitive medical information, as it complies with data privacy standards and builds trust among users. Additionally, Firebase offers high availability and minimal latency, which makes it suitable for real-time health monitoring even in large-scale or multi-user environments.

By integrating Firebase into this ESP8266-based health monitoring system, the project achieves a high level of performance, scalability, and reliability. The cloud platform eliminates the need for hosting and managing a dedicated backend server, simplifying system architecture and reducing deployment costs. Furthermore, Firebase can be seamlessly connected to visualization tools and dashboards, enabling health data to be displayed in intuitive formats such as charts, graphs, and logs. This makes it easier for users—whether they are patients, family members, or healthcare professionals—to understand health trends over time and make informed decisions.

Firebase, a cloud-based Backend-as-a-Service (BaaS) platform developed by Google, plays a crucial role in enabling real-time data storage and synchronization in the proposed health monitoring system. In this project, Firebase is used to store and manage the sensor data collected from the LM35 temperature sensor and the MPU6050 motion sensor. These sensors are connected to the ESP8266 microcontroller, which processes the data and uploads it to the Firebase Realtime Database via Wi-Fi. Firebase allows for continuous and automatic updates to the database, meaning that as soon as new sensor readings are captured, they are instantly available to users through a connected dashboard or web interface.

One of the key advantages of using Firebase is its ability to handle real-time synchronization across multiple devices. This ensures that healthcare providers, patients, or caregivers can access the most recent sensor data without delay, allowing for timely analysis and decision-making. For example, if the LM35 detects an abnormal rise in body temperature or the MPU6050 identifies sudden or unusual movement—such as a fall or tremor—the data is immediately uploaded to Firebase and reflected in the dashboard, helping to quickly assess potential health risks.

Moreover, Firebase supports structured data storage in JSON format, making it easy to organize and retrieve data based on different parameters such as timestamp, sensor type, or user ID. This makes the system scalable and adaptable for long-term health tracking or monitoring multiple users simultaneously. Firebase also provides built-in security features, allowing access control through authentication and rules, which is essential when dealing with sensitive health information.

The use of Firebase also reduces the need for a dedicated backend server, making the system more lightweight and cost-effective. It supports cross-platform integration, allowing the data to be visualized not only on web browsers but also on Android and iOS mobile apps. In this project, the dashboard fetches data from Firebase and displays it using graphical elements, making it easier for users to interpret trends in body temperature or motion over time.

2.4 Blynk App and Real-Time Data

The Blynk IoT platform is used specifically to handle real-time data from the MAX30100 sensor, which measures heart rate and blood oxygen saturation (SpO_2). Blynk is a powerful mobile-based IoT platform that enables developers to build interactive dashboards to monitor and control hardware remotely using smartphones.

The platform provides a user-friendly interface for visualizing sensor data and supports real-time updates, making it an ideal solution for health-related applications where immediate access to data is critical.

The MAX30100 sensor is connected to the ESP8266 microcontroller via the I2C communication protocol. As the ESP8266 collects data from the sensor, it transmits the heart rate and SpO_2 readings directly to the Blynk app over Wi-Fi. Within the Blynk mobile dashboard, widgets such as gauges, graphs, or value displays are configured to show the live readings, enabling users to monitor their cardiovascular health anytime and from anywhere. The responsiveness and simplicity of Blynk make it especially suitable for non-technical users, patients, or caregivers who need clear, real-time access to vital health metrics without dealing with complex interfaces.

Unlike the LM35 and MPU6050 sensors, whose data is sent to Firebase for storage and dashboard visualization, the MAX30100 data is processed separately through Blynk to reduce communication load and optimize the limited power and processing capacity of the ESP8266. Since all sensors use I2C and the ESP8266 has limited power output, splitting the data flow between Firebase and Blynk ensures more stable performance. This separation allows the system to maintain real-time responsiveness for heart rate and SpO_2 monitoring—two of the most time-sensitive parameters in health diagnostics.

Moreover, Blynk supports features such as data history, event notifications, and remote device control. These capabilities can be extended to alert users if their heart rate or SpO_2 levels fall outside safe ranges. This makes the platform not just a display tool, but also an early warning system for potential health issues. The real-time communication and visual feedback offered by Blynk greatly enhance the safety and usability of the system, especially in remote monitoring or telehealth scenarios. In this health monitoring system, the Blynk IoT platform is specifically implemented to manage and display real-time data from the MAX30100 sensor, which measures two critical physiological parameters: heart rate and blood oxygen saturation (SpO_2). Blynk serves as a mobile-first solution that allows remote monitoring of IoT-enabled devices through an intuitive and customizable app.

Chapter 3

SYSTEM DESIGN AND ARCHITECTURE

The health monitoring system is designed using the ESP8266 NodeMCU microcontroller, integrated with three sensors: MAX30100, LM35, and MPU6050. This setup enables the continuous monitoring of vital health parameters such as heart rate, SpO_2 (oxygen saturation), body temperature, and fall detection.

3.1 System Components

The proposed health monitoring system is designed using IoT technology to track vital health parameters such as heart rate, SpO_2 level, body temperature, and motion for fall detection. The system uses a combination of hardware components (sensors and microcontroller) and cloud platforms (Blynk and Firebase) to collect, store, and visualize data in real time. Below is a detailed explanation of the major components used in this system:

1. ESP8266 NodeMCU Development Board

The ESP8266 NodeMCU is the central control unit of the health monitoring system. It is based on the ESP8266 Wi-Fi SoC, offering GPIO, I2C, SPI, PWM, and ADC capabilities. In this project, it reads data from three sensors and communicates with two different cloud services—Firebase and Blynk—via Wi-Fi. The ESP8266's compact size and low power consumption make it ideal for portable health monitoring applications. However, it has only one ADC pin and limited I2C bandwidth, which is why data distribution between platforms is necessary to avoid overload. Using this microcontroller, real-time health data is captured and uploaded to cloud platforms for visualization and decision-making.

2. MAX30100 – Heart Rate and SpO_2 Sensor

The MAX30100 is a sophisticated biosensor that measures two key health parameters: heart rate and blood oxygen saturation (SpO_2). It uses two LEDs—red and infrared—and a photodetector to measure light absorption in the blood. Based on the level of light absorbed, it calculates pulse rate and SpO_2. These readings are crucial for early detection of cardiovascular or respiratory issues. In this system, data from the MAX30100 is transmitted directly to the Blynk IoT app, providing real-time monitoring on the user’s smartphone. Because of the need for instant updates and high sampling frequency, Blynk is preferred over Firebase for this sensor to maintain responsiveness.

3. LM35 – Analog Temperature Sensor

The LM35 is a precision analog temperature sensor with a linear output that varies directly with temperature in Celsius. It outputs 10 mV per °C, making it easy to convert voltage readings into temperature values using the ESP8266’s ADC pin. Body temperature is a fundamental health indicator, and abnormal readings can signify fever, infection, or other health problems. The data from the LM35 is sent to Firebase, where it is stored and later retrieved by the dashboard for visualization. Firebase allows this data to be continuously logged, which is useful for tracking trends over time or reviewing a patient's temperature history.

4. MPU6050 – Accelerometer and Gyroscope Sensor

The MPU6050 sensor is used in this project for fall detection and motion analysis. It combines a 3-axis accelerometer and a 3-axis gyroscope on a single chip, communicating via the I2C interface. Sudden or unusual changes in acceleration or angular velocity can indicate falls or erratic movement—especially useful in elderly health monitoring systems. The ESP8266 processes this motion data and sends it to Firebase, where it can be visualized and used to trigger alerts or emergency actions. Firebase can also be linked to an external notification system (e.g., SMS, email, or push notifications) if a fall is detected.

5. Breadboard

The breadboard is a solderless construction platform used to prototype and test the circuit. It enables easy mounting of the ESP8266 and sensors, as well as quick reconfiguration of the wiring. During the development phase, using a breadboard ensures flexibility and allows for easy debugging and hardware changes without permanent connections.

6. Jumper Wires

Jumper wires connect all the components—ESP8266, sensors, and power lines—on the breadboard. They come in male-to-male, female-to-male, and female-to-female configurations depending on the pin headers of the sensors and the microcontroller. Proper wiring is essential for clean signal transmission and stable sensor performance.

7. Blynk IoT Platform

Blynk is used as the real-time mobile interface in this system, particularly for the MAX30100 sensor. It allows users to see live heart rate and SpO_2 values on their smartphones. The app uses widgets such as value displays, graphs, and LEDs that can be customized in a drag-and-drop interface. Blynk's server handles the data sent from the ESP8266 and instantly updates the app interface. Because the MAX30100 requires fast, uninterrupted data transmission for real-time visibility, Blynk is an ideal choice, helping avoid data lag or loss common with heavier platforms.

8. Firebase Realtime Database

Firebase is used for cloud data storage and retrieval of the LM35 and MPU6050 sensor readings. It enables persistent storage of temperature and motion data, allowing dashboards or monitoring interfaces to access the information remotely. Firebase supports real-time updates, meaning as soon as new sensor data is uploaded, it's available across all connected clients.

9. Custom Dashboard or Web App (Optional Extension)

In addition to mobile monitoring through Blynk, a custom dashboard (e.g., made with HTML, JavaScript, and Firebase) can be created to visualize data from the LM35 and MPU6050 sensors. The dashboard can show charts, logs, and alerts using Firebase's real-time data feed.

Power Supply Consideration

All sensors use the I2C protocol and draw current from the ESP8266 board, which has a limited 3.3V output. Connecting all sensors at once can lead to power instability. To manage this, the system separates the sensor tasks: MAX30100 works with Blynk in real time. which is why data distribution between platforms is necessary to avoid overload. Using this microcontroller, real-time health data is captured and uploaded to cloud platforms for visualization and decision-making.

3.2 System Architecture

He proposed health monitoring system is designed using the ESP8266 NodeMCU microcontroller, integrated with three sensors: MAX30100, LM35, and MPU6050. This setup enables the continuous monitoring of vital health parameters such as heart rate, SpO_2 (oxygen saturation), body temperature, and fall detection. The system is designed to be compact, low-cost, and connected to the internet using Wi-Fi, which allows real-time monitoring and remote access to patient health data.

As shown in **Fig. 3.1**, the system architecture the MAX30100 sensor is responsible for measuring heart rate and SpO_2 levels. It operates on the I^2C communication protocol and transmits its data in real-time to the Blynk IoT platform. Blynk, a mobile-based application, is ideal for this purpose because it offers instant data visualization and fast user interface updates. This ensures that any critical changes in a user's heart rate or oxygen level are promptly available on a mobile dashboard for the user or healthcare provider.

On the other hand, the LM35 temperature sensor and MPU6050 motion sensor are used to detect body temperature and falls, respectively. The LM35 is an analog temperature sensor connected to the ESP8266's analog pin, while the MPU6050 (which includes an accelerometer and gyroscope) is connected via I^2C. These sensors send their readings to Google Firebase, a real-time cloud database. Firebase is used to store and retrieve data for longer-term tracking and dashboard visualization, enabling health professionals to analyze trends and receive alerts in case of abnormal activity.

Due to power limitations of the ESP8266 board—especially when multiple I^2C devices are connected—sensor data is routed selectively to different platforms to optimize performance and stability. The MAX30100 alone is connected to Blynk to reduce the real-time processing load and avoid Wi-Fi or I^2C bus failures. The LM35 and MPU6050 data, which are less frequent and not latency-sensitive, are sent to Firebase for background monitoring.

This dual-platform design ensures that the system balances both real-time alerts and long-term health tracking without overloading the ESP8266's capabilities. It also allows users to view real-time health updates on mobile devices while storing important sensor history securely in the cloud for future reference. The modular nature of the design also allows for future expansion.The entire system is powered by a 3.3V regulated power supply that supports the ESP8266 and all sensors. Since all the sensors rely on the same I²C bus, careful attention is given to avoid voltage drops and communication conflicts by using short wires and possibly dedicated power lines. The circuit is prototyped on a breadboard using jumper wires for easy testing and troubleshooting.

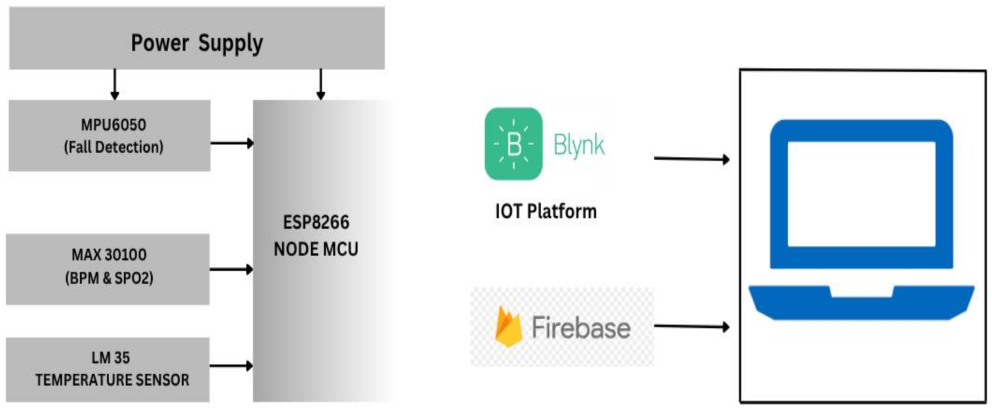


Fig. 3.1 Architecture of system

3.3 Data Flow and Processing

The health monitoring system that integrates multiple sensors with an IoT platform to collect and transmit vital health data. This system is designed to monitor important physiological parameters such as body temperature, heart rate, blood oxygen saturation, and fall detection in real-time. It is powered by a common power supply that energizes all components, including the sensors and the central microcontroller unit. The design focuses on continuous health monitoring with remote access capabilities using IoT technologies.

The data flow is explained in **Fig. 3.2** At the core of the system is the ESP8266 NodeMCU, a Wi-Fi-enabled microcontroller that serves as the main processing and communication unit. It collects data from three different sensors: MPU6050, MAX30100, and LM35. The ESP8266 not only reads the sensor data but also processes it to identify significant health indicators. Once the data is processed, it is transmitted wirelessly to cloud platforms and user interfaces, making it accessible to patients, caregivers, or healthcare professionals.

The MPU6050 sensor is responsible for fall detection. It contains a combination of an accelerometer and a gyroscope that detect rapid changes in orientation and motion. This makes it ideal for identifying situations where a person may have fallen. By analyzing the motion patterns, the system can differentiate between normal movements and potential falls, thus triggering alerts if necessary.

The MAX30100 sensor is a pulse oximeter and heart-rate monitor. It measures two key parameters: BPM (beats per minute) and SpO2 (blood oxygen saturation level). These metrics are crucial for understanding cardiovascular and respiratory health. The ESP8266 reads the optical signals from this sensor and converts them into readable heart rate and oxygen level values, which are then shared with the connected platforms. The LM35 sensor is a temperature sensor that provides analog voltage output proportional to the body temperature. It helps in tracking any signs of fever or abnormal body temperature. Since the sensor gives analog data, the ESP8266 processes this into readable temperature values. This simple but effective sensor plays a critical role in health diagnostics, especially for monitoring infections or fevers.

The processed data from the sensors is sent to Blynk, an IoT platform that provides real-time monitoring through a smartphone or web interface. Blynk enables the display of current sensor readings and can send immediate alerts or notifications when any health parameter crosses a dangerous threshold. This real-time aspect is vital in emergencies such as sudden drops in oxygen levels or accidental falls. Simultaneously, the system uses Firebase, a cloud-based database, to store the health data over time. Firebase enables the system to maintain a historical log of all sensor readings, which can be used for long-term analysis or remote access by healthcare providers. This ensures that medical professionals or family members can view the patient's health trends, even if they are not physically present.

In the end, the collected data is made accessible to the end-users via PCs or smartphones. This output interface ensures that the information is easy to interpret and can be acted upon promptly. The system architecture supports both live monitoring and historical data analysis, making it a comprehensive solution for personal and remote healthcare monitoring. this health monitoring system effectively combines various biomedical sensors with an IoT-enabled microcontroller to collect, process, and transmit health data. The integration with platforms like Blynk and Firebase enables real-time alerts and long-term data storage, making the solution ideal for elderly care, chronic disease monitoring, and post-operative recovery. Its design reflects the potential of modern IoT systems to improve healthcare accessibility and responsiveness.

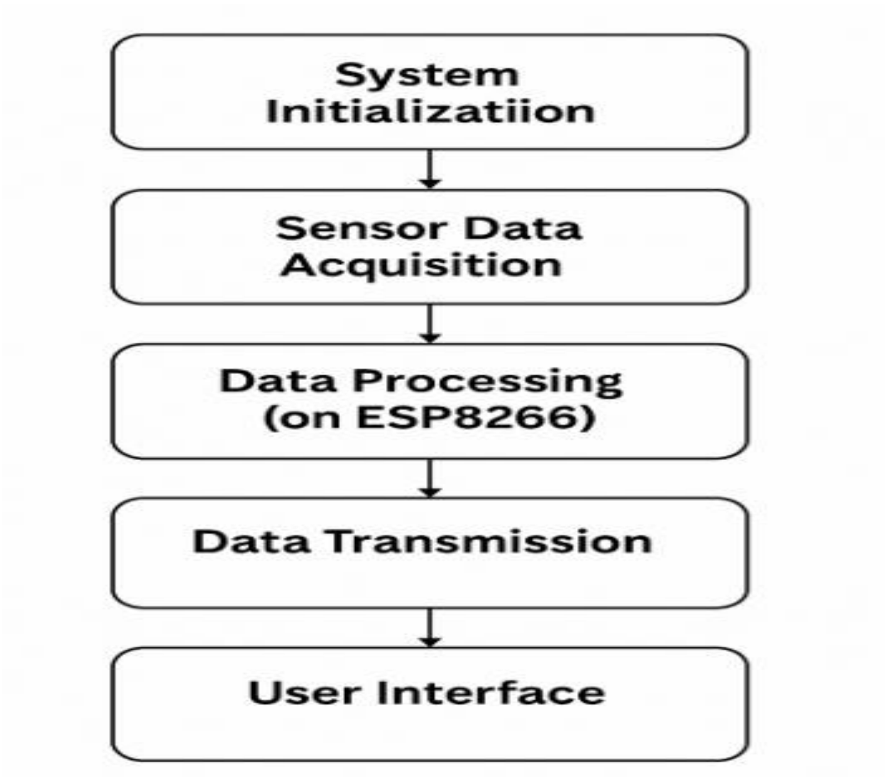


Fig. 3.2 Data Flow and Processing

3.4 Modules

1. ESP8266 NodeMCU

The ESP8266 NodeMCU is a versatile microcontroller module that plays a vital role in modern Internet of Things (IoT) applications, including portable health monitoring systems. It is built around the ESP8266 Wi-Fi System-on-Chip (SoC), which offers integrated TCP/IP protocol stack, enabling seamless Wi-Fi connectivity. In this project, the ESP8266 acts as the brain of the system, coordinating sensor data collection and communication with remote cloud platforms. Its 32-bit Tensilica L106 microcontroller core, running at 80 or 160 MHz, provides enough computational power for handling real-time sensor inputs while maintaining energy efficiency—an essential factor for battery-operated health monitoring devices.

The ESP 8266, responsible for Hardware Setup, is shown in Fig. 3.3 To distribute the data load and avoid overburdening a single cloud service, the ESP8266 transmits data to two platforms: Firebase and Blynk. Firebase offers scalable real-time database storage and synchronization capabilities, which is ideal for logging health data for historical analysis, trend detection, and cloud-side computation. Meanwhile, Blynk provides a user-friendly mobile interface for real-time monitoring, allowing users to view health metrics instantly via a smartphone app. This dual-platform approach ensures a balance between real-time responsiveness and long-term data archiving.

The integration of these technologies transforms the ESP8266 NodeMCU into an efficient IoT gateway for health data. The microcontroller captures physiological signals, processes them locally, and transmits the data wirelessly. This enables both patients and healthcare providers to monitor health parameters remotely, reducing the need for frequent in-person consultations and enabling proactive healthcare. Its small form factor also allows it to be embedded in wearable or portable medical devices, enhancing usability and patient comfort.

Overall, the ESP8266's affordability, built-in Wi-Fi, and support for various communication protocols make it an excellent choice for connected health applications. Although its hardware limitations require workarounds, its compatibility with a wide array of sensors and cloud services makes it a powerful platform for building real-time, portable, and scalable health monitoring systems. Its small form factor also allows it to be embedded in wearable or portable medical devices, enhancing usability and patient comfort.

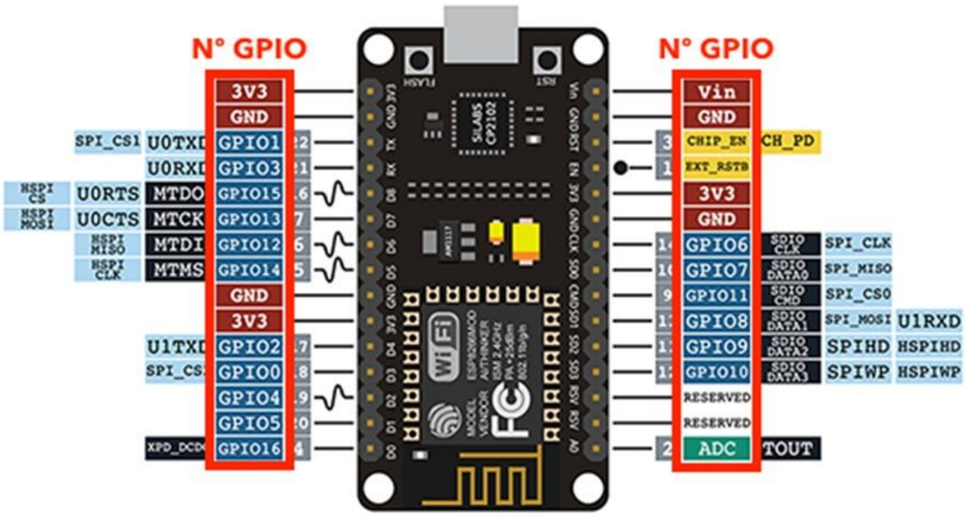


Fig. 3.3 NodeMCU ESP8266

2. Max 30100 Sensor

The MAX30100 is a highly integrated pulse oximetry and heart-rate monitor module that is particularly well-suited for wearable health applications. It combines two essential optical sensors—one for red light (660 nm) and one for infrared light (880 nm)—alongside a photodetector, low-noise analog signal processing, and digital signal processing on a single chip. These components work in unison to detect changes in blood volume in the microvascular bed of tissue.

The difference in light absorption at two wavelengths allows the MAX30100 to calculate SpO_2 (oxygen saturation) and heart rate with a reasonable degree of accuracy. To perform its measurements, the MAX30100 emits pulses of red and infrared light into the skin (usually the fingertip or earlobe), and the photodetector senses the amount of light that is either transmitted through or reflected by the tissues. Oxygenated and deoxygenated hemoglobin absorb these wavelengths differently—oxygenated hemoglobin absorbs more infrared light, while deoxygenated hemoglobin absorbs more red light. By analyzing the ratio of light absorbed at each wavelength, the chip estimates the percentage of oxygen in the blood (SpO_2). Heart rate is determined by tracking the periodic fluctuations in absorption, which correspond to heartbeats.

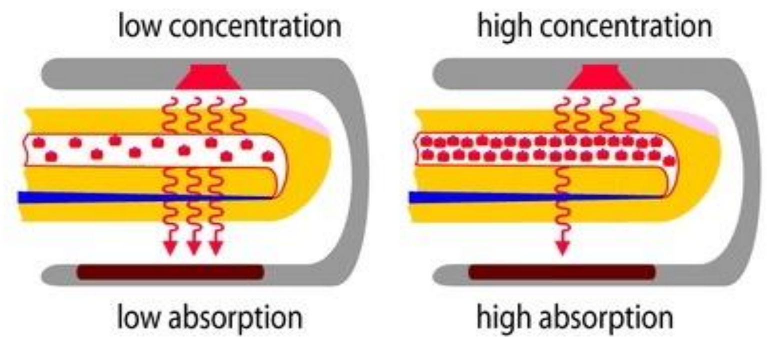


Fig. 3.4 Working of MAX30100 Sensor

In this system, data from the MAX30100 is prioritized for real-time transmission via the Blynk IoT platform, which offers low-latency communication and user-friendly mobile interface capabilities. Blynk allows for immediate visual feedback on the user's smartphone through widgets like real-time gauges, charts, and notifications. This is critical for vital signs like heart rate and oxygen saturation, which can change rapidly and may require prompt medical attention. Firebase, while excellent for data logging and analysis, has slightly higher latency and is more suited to batch uploads or historical tracking, making Blynk the ideal choice for this real-time application. To perform its measurements, the MAX30100 emits pulses of red and infrared light into the skin (usually the fingertip or earlobe), and the photodetector senses the amount of light that is either transmitted through or reflected by the tissues. Oxygenated and deoxygenated hemoglobin absorb these wavelengths differently—oxygenated hemoglobin absorbs more infrared light, while deoxygenated hemoglobin absorbs more red light.

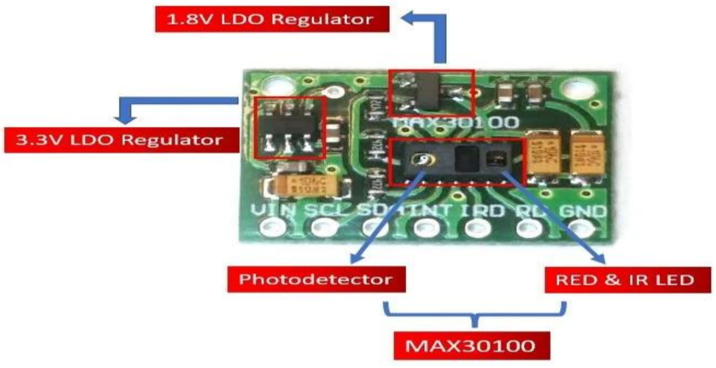


Fig. 3.5 MAX30100 Sensor

The Max30100 sensor, responsible for BPM & SPO2, is shown in **Fig. 3.5** additionally, the MAX30100's relatively high sampling rate and sensitivity to movement and ambient light require careful signal processing to ensure reliable readings. In many systems, a low-pass filter or averaging algorithm is applied on the ESP8266 before sending the data to Blynk. This helps eliminate noise and false spikes due to motion artifacts or poor finger placement. By continuously streaming this filtered data to Blynk, users can not only monitor their health in real time but also configure alarms for abnormal readings, enhancing safety and preventive care.

Overall, the MAX30100's compact size, low power consumption, and dual-functionality make it an excellent biosensor for portable and wearable health monitoring systems. Its integration with the ESP8266 and Blynk creates a robust, user-friendly solution that empowers individuals to take a more proactive role in managing their cardiovascular and respiratory health.

3. LM35 Sensor

The LM35 sensor, responsible for temperature, is shown in **Fig. 3.6** the LM35 is a widely used analog temperature sensor known for its simplicity, accuracy, and cost-effectiveness. Designed to measure temperature in degrees Celsius, it produces a linear output of 10 millivolts per °C, which simplifies the conversion of analog voltage readings into temperature values without the need for complex calibration or signal conditioning. This makes it ideal for integration with microcontrollers like the ESP8266, which can read analog signals through its single ADC (Analog-to-Digital Converter) pin. In the context of a health monitoring system, the LM35 is used to measure body temperature, a critical vital sign for detecting infections, fever, or metabolic disturbances.

When connected to the ESP8266, the LM35 provides continuous analog voltage proportional to the body temperature. The ESP8266 reads this voltage, converts it into a temperature value using a simple formula (Temperature = Voltage in mV / 10), and prepares it for transmission. However, since the ESP8266 has only one ADC pin, care must be taken to ensure no conflict arises when using multiple analog sensors. In many implementations, the LM35 is read intermittently rather than continuously to conserve power and reduce ADC overload, especially when the ESP8266 is already handling multiple tasks or sensors.

In this system, Firebase is used as the cloud backend for storing and retrieving temperature data collected from the LM35. Firebase offers real-time database capabilities that allow data to be logged, timestamped, and accessed remotely through a dashboard interface. Unlike real-time display apps like Blynk, Firebase is better suited for long-term data logging and trend analysis. Healthcare providers or patients can review past temperature readings, which helps in tracking the progression of fevers or monitoring recovery from illnesses. monitoring, or personal health tracking applications.

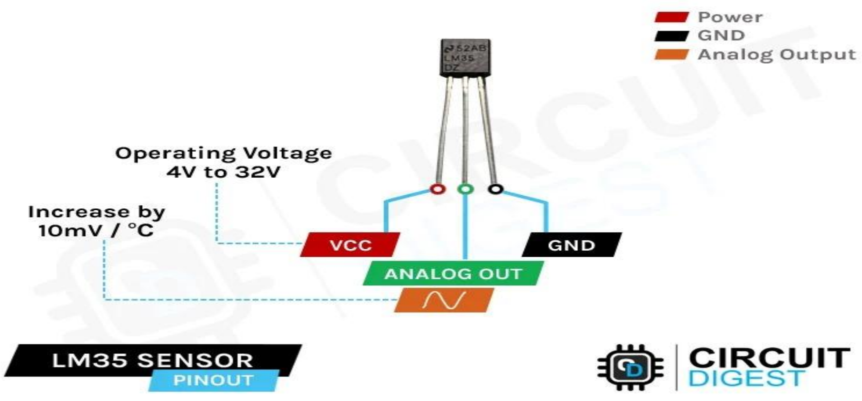


Fig. 3.6 LM35 Temperature Sensor

The advantage of logging LM35 data to Firebase lies in its scalability and ability to integrate with web or mobile dashboards. These dashboards can visualize temperature trends over hours, days, or weeks using charts and graphs. Alerts can also be programmed to notify users or caregivers if the temperature crosses a predefined threshold, indicating a potential health issue that requires attention. Moreover, since Firebase supports data synchronization across multiple devices, it enables healthcare professionals to monitor patients remotely and in real-time.

3. MPU6050 Sensor

The MPU6050 sensor, responsible for fall detection, is shown in Fig. 3.7 the MPU6050 is a compact and powerful motion sensor that combines a 3-axis accelerometer and a 3-axis gyroscope in a single integrated circuit, making it a 6-degree-of-freedom (6-DOF) inertial measurement unit (IMU). It is widely used in applications ranging from robotics and mobile devices to health monitoring systems due to its low cost, small size, and high precision. The sensor communicates with microcontrollers like the ESP8266 through the I2C protocol, allowing easy integration with minimal wiring. It also includes an internal Digital Motion Processor (DMP) that can perform complex calculations such as sensor fusion and orientation estimation, reducing the processing load on the microcontroller.

In the context of health monitoring, the MPU6050 is valuable for tracking movement, orientation, and acceleration of the human body. It can detect activities like walking, sitting, running, or lying down by analyzing acceleration patterns. Additionally, it is commonly used for fall detection, which is critical for elderly care systems. The gyroscope helps monitor rotational motion, which, when combined with accelerometer data, can give a full picture of body orientation and posture. This makes the MPU6050 ideal for posture correction devices, physical therapy tracking, and sleep monitoring systems.

When connected to the ESP8266, the MPU6050 sends real-time motion data, which can either be processed locally or transmitted to cloud platforms like Firebase for long-term storage and analysis, or to Blynk for immediate visualization on a mobile app. Due to its high sampling rate and the volume of data it generates, using the DMP feature is recommended to pre-process the data and reduce communication overhead. This helps maintain efficient system performance, especially when multiple sensors are in use.Despite its advantages, the MPU6050 does have some limitations. It does not include a magnetometer, so it cannot determine absolute orientation or heading without additional sensors. It also shares the I2C bus with other devices, so careful management is required to avoid communication delays. Nonetheless, the MPU6050 remains a reliable and efficient solution for capturing motion-related health data, enabling a wide range of smart health monitoring applications.

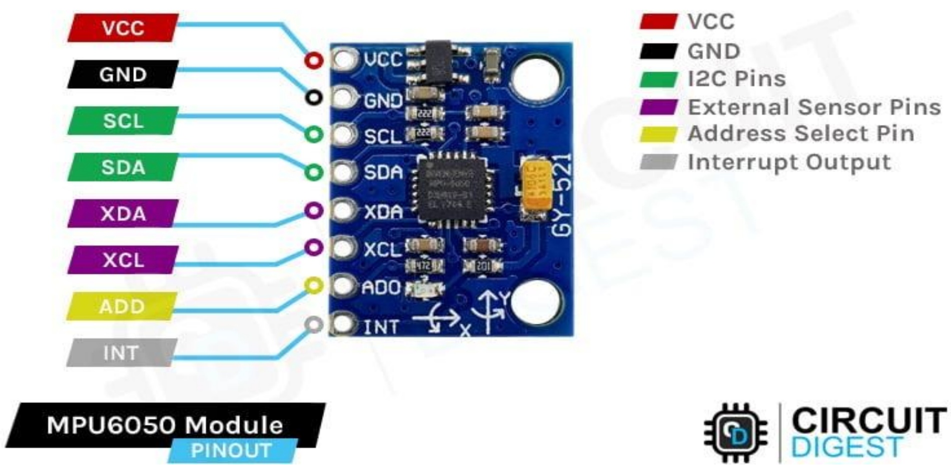


Fig. 3.7 MPU6050 Sensor

5. Breadboard

The breadboard is a crucial prototyping tool widely used in IoT-based health monitoring systems for building and testing electronic circuits without soldering. It allows components such as microcontrollers (like the ESP8266), sensors (e.g., LM35, MAX30100, MPU6050), resistors, and jumper wires to be easily connected and rearranged. In the context of health monitoring, the breadboard plays an important role during the development and testing phase, where sensor configurations, wiring, and signal processing circuits are frequently adjusted before final deployment.For IoT health systems, the breadboard enables quick integration of multiple sensors that monitor parameters such as body temperature, heart rate, blood oxygen levels, and motion.

The Breadboard, responsible for connecting wires, is shown in **Fig. 3.8.** It allows developers to test the interaction between these components and the microcontroller, ensuring reliable data acquisition and communication with cloud platforms like Firebase or Blynk. For instance, the LM35 sensor can be inserted into the breadboard and connected directly to the ESP8266's ADC pin, while the

MAX30100 and MPU6050 can be connected via the I2C interface, with pull-up resistors also easily placed on the breadboard if needed.One of the key advantages of using a breadboard in an IoT health project is flexibility. Developers can test different power supply arrangements, add filtering components like capacitors to stabilize signals, and easily replace faulty or poorly performing sensors. It also supports rapid troubleshooting since connections are visible and adjustable, making it easier to debug issues like sensor misreadings or communication errors. Additionally, when prototyping wearable or portable health systems, the breadboard helps determine the best sensor layout and wiring configuration before moving to a compact, soldered PCB design.

However, breadboards are typically used only during the development phase. For real-world deployment, especially in wearable or patient-focused devices, the system needs to be transferred to a more robust and compact form—such as a custom PCB or perfboard—to ensure stability, durability, and safety. Breadboards can be affected by loose connections, interference, and limited current capacity, which is not suitable for long-term use in healthcare environments.

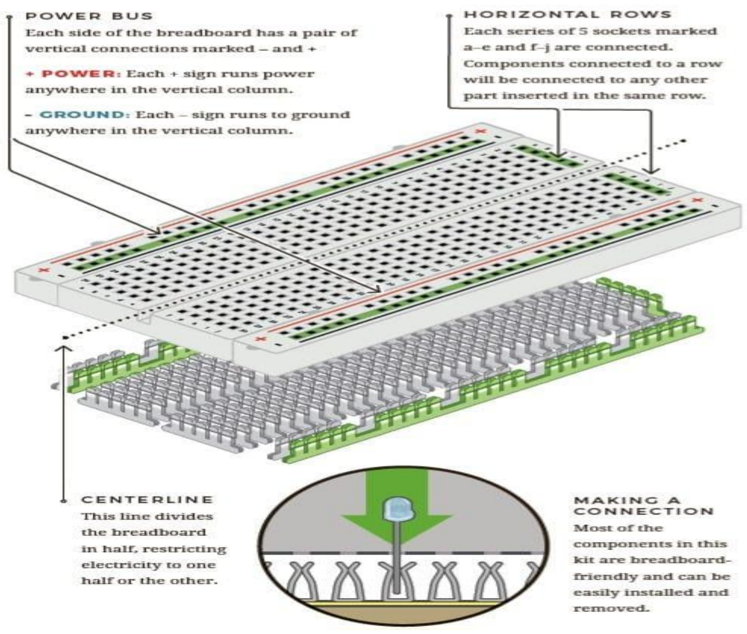

POWER BUS
Each side of the breadboard has a pair of vertical connections marked – and +
+ POWER: Each + sign runs power anywhere in the vertical column.
– GROUND: Each – sign runs to ground anywhere in the vertical column.
HORIZONTAL ROWS
Each series of 5 sockets marked a–e and f–j are connected. Components connected to a row will be connected to any other part inserted in the same row.
CENTERLINE
This line divides the breadboard in half, restricting electricity to one half or the other.
MAKING A CONNECTION
Most of the components in this kit are breadboard-friendly and can be easily installed and removed.

Fig. 3.8 Bread Board

6. Jumper Wires

The Jumper wires, responsible for connecting the hardware component, is shown in **Fig. 3.9** Jumper wires, also known as jumper wires, are essential components in IoT circuit prototyping, especially when building health monitoring systems. These wires are used to make electrical connections between various components on a breadboard or between a breadboard and a microcontroller like the ESP8266. They are available in male-to-male, male-to-female, and female-to-female types, depending on the connection points needed. Jumper wires simplify the process of creating temporary or semi-permanent circuits without the need for soldering, making them ideal for testing and development.

In an IoT-based health monitoring system, jumper wires are used to connect sensors—such as the LM35 temperature sensor, MAX30100 pulse oximeter, and MPU6050 motion sensor—to the ESP8266 NodeMCU or another controller. For example, the analog output of the LM35 can be connected to the ESP8266's ADC pin using a jumper wire, while I2C-based sensors like the MPU6050 use jumper wires to connect their SDA and SCL lines to the respective I2C pins on the microcontroller. Power (VCC) and ground (GND) connections are also made using jumper wires, ensuring proper operation of all components in the circuit.

Jumper wires offer high flexibility and reusability, which is particularly important in prototyping IoT systems where sensor positions and wiring configurations often change. They allow quick reconfiguration of circuit layouts, making it easier to test different hardware setups or troubleshoot issues like incorrect readings or unstable connections.

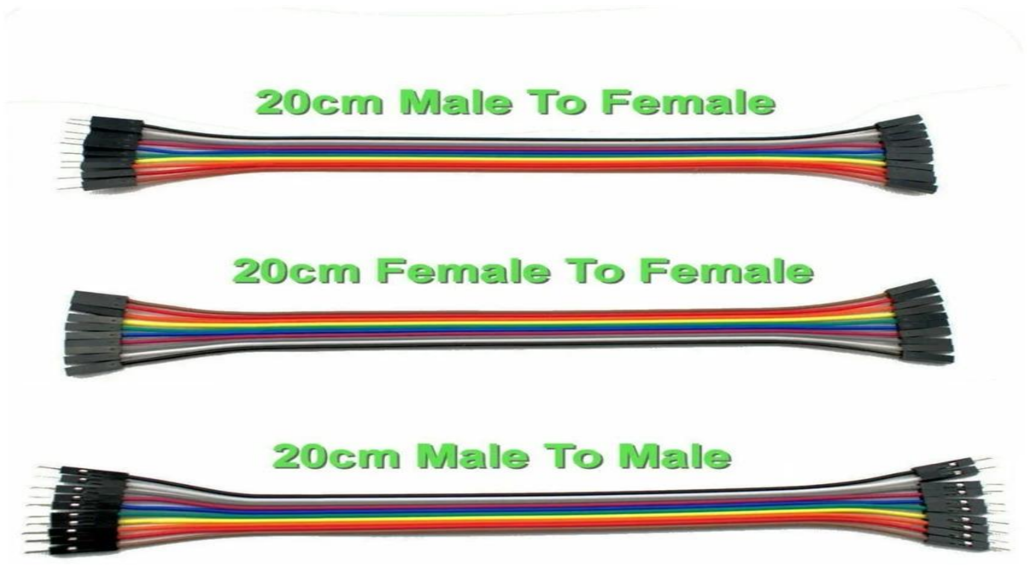


Fig. 3.9 Jumper Wires

3.5 User Interface Design

As shown in **Fig. 3.10**, The user interface (UI) of the IoT-based health monitoring system is a crucial component that bridges the gap between complex sensor data and actionable insights for healthcare professionals, caregivers, and patients. The design of the interface ensures real-time visibility into key physiological parameters such as body temperature, heart rate, blood oxygen level (SpO_2), and body motion, which are continuously monitored by a suite of sensors connected to an ESP8266 NodeMCU microcontroller. The web-based dashboard, built using HTML, CSS, and JavaScript, is the central visualization platform and allows for data interpretation in an intuitive, user-friendly manner. It retrieves sensor data from Firebase in real-time and updates visual components dynamically, ensuring users are always presented with the most recent health information.

The layout of the dashboard has been structured for clarity and ease of use. It features a left-hand sidebar with navigational links such as "Home," "History," and "About Us." These links help users quickly access different sections of the system. The “History” button is particularly useful as it enables healthcare providers to download and review past data, which is essential for long-term monitoring and trend analysis. The main section of the dashboard begins with a prominent title, "Health Monitor System," and a live timestamp that indicates the last time data was updated from the sensors. This feature is crucial in health monitoring scenarios where timing is critical. If the data is outdated or not updating, users are immediately made aware through the absence of a recent timestamp.

A standout element of the UI is the status box panel, which displays real-time numeric values for each monitored health parameter. These include body temperature (from the LM35 sensor), heart rate and SpO_2 (from the MAX30100 sensor), and motion data (from the MPU6050 accelerometer/gyroscope sensor). There is also a status indicator for fall detection, which uses acceleration thresholds to determine if a sudden fall has occurred. Each of these readings is displayed in its dedicated box, labeled and color-coded for quick recognition. This modular design not only improves readability but also allows for scalability—new sensor data types can be added without significantly altering the interface layout.

To complement the numeric display, the UI features interactive data visualizations using Chart.js, a powerful JavaScript library. Graphs are provided for each of the acceleration axes (X, Y, Z), as well as for temperature trends. These charts update in real-time and provide a continuous timeline of the recorded data, helping users observe fluctuations and patterns. This graphical representation is invaluable in medical decision-making. For example, repeated sharp spikes in acceleration could indicate tremors or seizures, while a steady increase in temperature might point to fever or infection. By presenting this data visually, the system aids in quicker comprehension and more accurate assessments, especially in fast-paced clinical environments.

The UI also includes an alert box system, designed to grab the user's attention in case of abnormal sensor readings. When a reading goes beyond a safe threshold, a visually distinct alert box appears with a warning message and an icon indicating the nature of the issue. This real-time alert system is vital in scenarios such as detecting dangerously high heart rates or falls in elderly patients. Instead of requiring constant user vigilance, the system takes on a proactive role, notifying users only when immediate action may be needed. This feature significantly enhances the usability and safety of the platform, especially when monitoring patients remotely or at night.

Another important aspect of the UI is its responsive and scalable architecture. Using CSS classes such as "full" and "half" for chart containers, the layout adapts to various screen sizes and devices, making the dashboard accessible from desktops, laptops, and even tablets. This flexibility is critical in healthcare, where medical personnel may access the system from different locations or on the go. Additionally, the modular design approach allows developers to update or expand the system without overhauling the entire interface. For instance, integrating a new sensor like a blood pressure monitor would simply require adding a new status box and corresponding chart, without affecting existing components.

In summary, the user interface of your IoT-based health monitoring system is not just an aesthetic or organizational element—it is the core medium through which users interact with complex sensor data. It successfully translates raw, real-time physiological signals into readable formats that support informed medical decision-making. This real-time alert system is vital in scenarios such as detecting dangerously high heart rates or falls in elderly patients. Instead of requiring constant user vigilance, the system takes on a proactive role, notifying users only when immediate action may be needed. This feature significantly enhances the usability and safety of the platform, especially when monitoring patients remotely or at night.

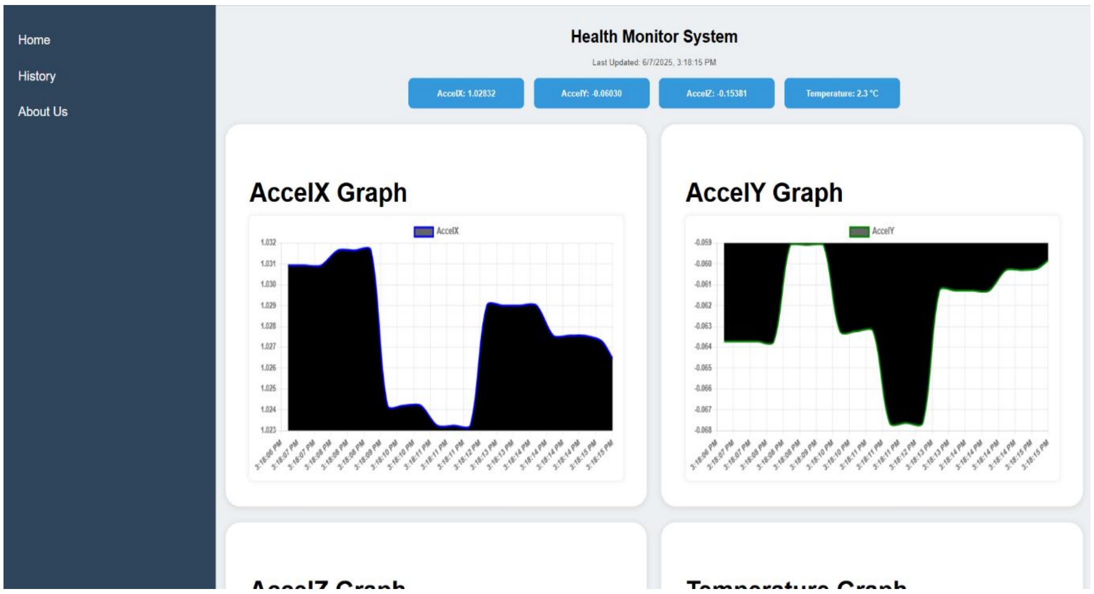


Figure. 3.10 User Interface.

3.5 Alert Mechanisms

In this IoT-based health monitoring system, the MAX30100 sensor plays a critical role in measuring two vital physiological parameters: heart rate and blood oxygen saturation (SpO_2). This sensor module integrates a red LED, an infrared LED, and a photodetector to measure the intensity of light absorbed by the blood, which varies with the volume of blood and its oxygen content. The sensor outputs pulse and oxygen saturation data by analyzing the periodic changes in the light absorption caused by heartbeats. The MAX30100 operates on low power and provides highly accurate readings, making it suitable for continuous health monitoring in portable or wearable devices. Its small size, reliability, and dual-sensing capabilities make it an essential part of this health monitoring system.

The data from the MAX30100 is collected by the ESP8266 NodeMCU, which processes the sensor readings and sends them to the Blynk IoT platform. Blynk is used for real-time visualization and interaction with IoT devices and is particularly well-suited for applications requiring mobile-based monitoring. In this system, Blynk receives the live heart rate and SpO_2 data and displays it on the user's smartphone using gauge widgets and numeric displays. The interface is clean and user-friendly, allowing patients and healthcare professionals.

To enhance patient safety and enable rapid response to health risks, the system includes an automated alert mechanism** based on predefined thresholds for both heart rate and SpO_2 levels. These thresholds are configurable and reflect medically acceptable ranges—for example, a heart rate between 60–100 beats per minute and an SpO_2 level above 95% are considered normal for a healthy adult. If the readings fall below or rise above these safe thresholds—such as in cases of bradycardia, tachycardia, or hypoxemia—the system triggers an alert through the Blynk platform. This is done by using Blynk Events, a feature that enables condition-based triggers and responses directly from the mobile app or cloud platform.

The Blynk Event feature is used to define conditions such as "if heart rate < 60 bpm" or "$SpO_2 < 90\%$" and associate them with automated actions. When one of these conditions is met, an event is triggered that executes a pre-configured notification action. In this project, the selected action is to send an alert message to the registered Gmail ID of the patient. This is achieved by linking the Blynk event to an automation service (such as IFTTT or Blynk's built-in email functionality) which takes the triggered event and sends an email containing the alert message and sensor readings. This ensures that the patient or their caregiver is immediately informed of any health irregularities, even if they are not actively checking the Blynk app.

The email message contains essential details such as the nature of the alert (e.g., "Heart Rate Too Low"), the current sensor value, the time the event was triggered, and a suggested action such as consulting a doctor or checking the patient's condition. By leveraging email alerts, the system bridges the gap between real-time monitoring and proactive healthcare intervention. Unlike push notifications, which require the user to be actively using their phone or app, email alerts provide a more reliable channel that is likely to reach the user in time, regardless of their current activity. This approach significantly enhances the system's effectiveness in remote health management, especially for elderly users or patients with chronic conditions.

Chapter 4

IMPLEMENTATION

This system is designed to continuously monitor key physiological parameters, including body temperature, heart rate, oxygen saturation (SpO2), and fall detection, using three primary sensors: the LM35 temperature sensor, the MAX30100 pulse oximeter, and the MPU6050 accelerometer-gyroscope module.

4.1 Hardware Setup

The proposed health monitoring system leverages the ESP8266 NodeMCU microcontroller, a compact and cost-effective development board with built-in Wi-Fi, making it ideal for Internet of Things (IoT) applications. This system is designed to continuously monitor key physiological parameters, including body temperature, heart rate, oxygen saturation (SpO2), and fall detection, using three primary sensors: the LM35 temperature sensor, the MAX30100 pulse oximeter, and the MPU6050 accelerometer-gyroscope module.

As shown in **Fig. 4.1**, system setup It is an I2C-compatible sensor that combines pulse oximetry and heart-rate sensing functions into a single module. The sensor is interfaced with the ESP8266 using digital pins D2 (SDA) and D1 (SCL). Its power pins (VCC and GND) are connected to the 3.3V and ground of the ESP8266 respectively. This sensor sends real-time data to the Blynk IoT platform, allowing users to view their BPM and SpO2 readings through a mobile application.

The LM35 sensor is used to detect the user's body temperature. Unlike the other two, the LM35 outputs analog signals, which are read using the A0 analog pin of the ESP8266. This sensor is powered using the 3.3V pin of the NodeMCU and grounded through a common GND pin. Temperature data is pushed to the Firebase cloud database, making it accessible on web dashboards for remote monitoring and data logging purposes.

The MPU6050 module detects motion and orientation changes, enabling fall detection. It combines a 3-axis accelerometer and a 3-axis gyroscope and communicates using the I2C protocol, sharing the same SDA and SCL lines (D2 and D1) with the MAX30100 sensor. However, due to power constraints and I2C bus conflicts when all sensors are active simultaneously, a division of tasks is implemented to reduce load and ensure data accuracy.

To address these hardware limitations, data from the MAX30100 is solely transmitted to the Blynk platform, while data from the LM35 and MPU6050 are routed to Firebase. This modular communication strategy optimizes data flow and avoids I2C interference, particularly when multiple sensors attempt to communicate over the same lines simultaneously. This design choice enhances system stability and allows for smooth, real-time data handling.

The ESP8266's built-in Wi-Fi module facilitates wireless data transmission to both Blynk and Firebase platforms. By using cloud-based storage and visualization tools, the system ensures continuous data availability for remote users and healthcare professionals. The Firebase platform provides robust real-time database functionality, while Blynk offers an intuitive mobile interface for immediate health status updates. A regulated power supply is used to ensure stable and consistent voltage delivery to all components. Since all sensors are sensitive to voltage fluctuations—especially the MAX30100 and MPU6050 operating at 3.3V—using a properly regulated power source is critical. Inadequate power can result in inaccurate readings or malfunctioning components, which may compromise the system's reliability.

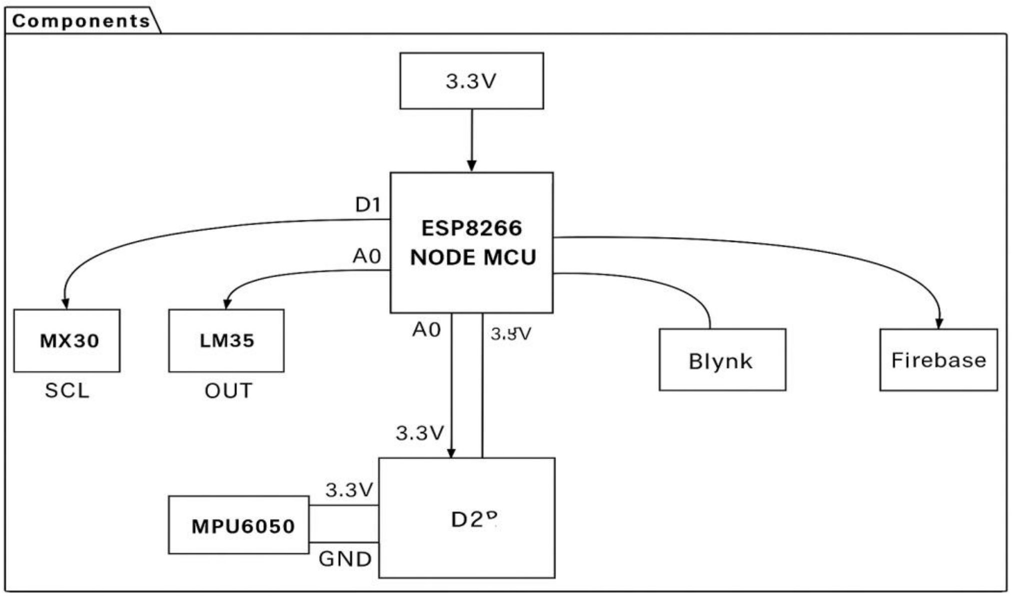


Fig. 4.1 System Setup.

4.2 Software Development

The system depicted in the diagram integrates multiple sensors with an ESP8266 Node MCU microcontroller to create a comprehensive health monitoring and fall detection device. The software setup involves several key components, including firmware programming for the microcontroller, configuring the sensors, setting up cloud-based IoT platforms, and ensuring seamless data communication and visualization. The core microcontroller, ESP8266 Node MCU, serves as the central processing unit, interfacing with the MPU6050 accelerometer and gyroscope for fall detection, the MAX30100 sensor for measuring BPM (beats per minute) and SpO2 (oxygen saturation), and the LM35 temperature sensor.

As shown in **Fig. 4.2**, the workflow the first step in the software setup involves writing the firmware for the ESP8266 Node MCU using a development environment like the Arduino IDE or Platform IO. The firmware must initialize each sensor module through their respective communication protocols: I2C for MPU6050 and MAX30100, and analog input for the LM35 temperature sensor. Careful calibration routines need to be implemented to ensure accurate data acquisition, particularly for physiological parameters that require precise readings. Additionally, the MPU6050 sensor data requires processing algorithms such as threshold-based detection or machine learning models embedded in the firmware to reliably detect fall events.

Once the sensor data is collected and processed locally, the next critical software step is enabling wireless communication between the ESP8266 and the cloud-based IoT platforms. The ESP8266 is equipped with Wi-Fi capabilities, which must be configured to connect to a local wireless network. After network setup, the firmware should implement communication protocols such as HTTP, MQTT, or REST APIs to send the sensor data securely to the IoT platforms — in this case, Blynk and Firebase. These platforms facilitate remote monitoring and data storage, enabling users to access real-time health metrics and alerts via a web or mobile interface.

Blynk serves as an IoT application development platform that provides an easy-to-use dashboard interface for real-time data visualization and control. The software setup for Blynk involves creating a project on the Blynk app, generating authentication tokens, and embedding these tokens within the ESP8266 firmware to authenticate device communications. Data sent to Blynk can then be displayed in the form of gauges, graphs, or notifications, providing an intuitive way to monitor parameters like heart rate, blood oxygen levels, temperature, and fall incidents remotely.

Firebase, on the other hand, is used as a backend database and cloud service provider. The software setup requires configuring Firebase Realtime Database or Fire store to store continuous sensor data streams for historical analysis and event logging. The firmware includes Firebase SDK integration, allowing the Node MCU to push data updates to Firebase. Firebase also supports cloud functions and authentication services, which can be leveraged to trigger automated alerts, store user profiles, or analyze trends over time, enhancing the system's overall functionality and reliability.

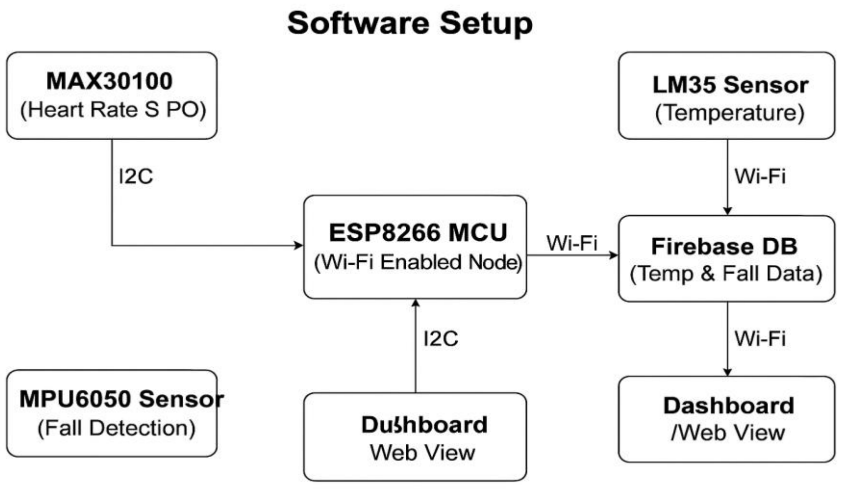


Fig. 4.2 Workflow.

4.3 Testing and Validation

The testing and validation phase of our IoT-based health monitoring system, built using the ESP8266 NodeMCU and sensors like the LM35, MAX30100, and MPU6050, was designed to ensure the system’s performance, reliability, and security in real-world scenarios. Initial tests began with evaluating the IoT device's connectivity under different Wi-Fi conditions. The ESP8266 was placed at varying distances from the router, from within the same room to around 10 meters away with one wall in between. The network had a typical home broadband speed of around 50 Mbps. Under these conditions, the ESP8266 consistently established a connection within 5 to 10 seconds of power-up. Even when the Wi-Fi was deliberately turned off and back on, the device reconnected within 30 seconds automatically without requiring a manual reset, thus demonstrating its reliability as an IoT node. A built-in LED on GPIO2 was programmed to reflect connection status, blinking during reconnection attempts and staying solid once a successful link was established.

A major component of our validation involved real-time data streaming to Firebase. Each sensor reading—body temperature from LM35, heart rate and SpO_2 from MAX30100, and accelerometer values from MPU6050—was updated to the Firebase Realtime Database every 15 seconds. The path `/sensor_data/current_readings` was continuously monitored to ensure timely updates and correct JSON formatting. Throughout our tests, data transmission remained consistent, with no packet loss or corrupted entries. The use of HTTPS for communication between the ESP8266 and Firebase ensured that all data was encrypted using TLS/SSL, maintaining the confidentiality and integrity of sensitive health data during transmission over the internet. These tests confirmed the system's reliability for secure, real-time medical telemetry.

In parallel, the web application and Firebase alerting system were rigorously tested for performance and responsiveness. The system was programmed to trigger alerts when predefined thresholds were crossed—such as a fall detected by the MPU6050 or abnormal heart rate/temperature levels. These alerts were recorded in Firebase under `/alerts/latest_alert_code` and `/alerts/latest_alert_message`. Testing showed that alerts were logged within 1 to 2 seconds after the event occurred. These changes were immediately reflected on the web dashboard, where medical professionals could view the alerts in real-time. The alert history was also appended with Firebase server timestamps, which helped in creating an audit trail. Importantly, only authenticated users with verified credentials could access these alerts, ensuring that sensitive health data was only visible to authorized personnel.

Sensor accuracy and edge-side processing were also a core focus during validation. The MAX30100 sensor was compared with a commercial-grade pulse oximeter. After allowing 10-15 seconds for stable finger placement, the heart rate readings were within ±5 BPM, and the SpO_2 values were within ±2% of the reference device. Tests highlighted that good finger placement and minimal movement were critical for accurate readings; otherwise, the device would show errors or zero values, triggering an alert to wait for valid data. The LM35 temperature sensor was cross-verified using a digital clinical thermometer. When held against the skin, the sensor responded within 1-2 seconds and consistently reported values in the 36–37°C range with a ±0.5°C margin. These accurate readings proved the viability of our system for continuous body temperature monitoring.

The MPU6050 accelerometer was validated through multiple movement simulations to test the fall detection logic. In a stationary state, it recorded \~1.0G. Small movements produced slight variations, while rapid motion or simulated falls generated spikes exceeding 2.5G. Out of 20 drop tests from approximately 1 meter onto a cushioned surface, 18 events were successfully classified as falls by the system. The two missed detections occurred when the device bounced, which disrupted the 'no-motion' condition that confirms a fall. No false positives were recorded during normal usage. An initial threshold of 2.0G led to frequent false alerts, so it was increased to 2.5G for better precision. This iterative calibration improved fall detection accuracy and minimized false triggers.

The security and usability of the web application were evaluated by simulating access attempts using both valid and invalid credentials. Only users with approved medical login details were granted access to the dashboard. The login mechanism effectively blocked all unauthorized users, displaying relevant error messages and logging failed attempts. The interface itself was designed to be user-friendly for medical professionals, presenting a consolidated view of all vitals with easy-to-read charts and clear alert summaries. Moreover, the web application was tested on various devices, including desktops, tablets, and smartphones. The interface adjusted smoothly to different screen sizes, ensuring that the system was accessible for medical teams on-the-go, regardless of device used.

To evaluate the complete system's performance under extended operation, the device was run continuously for several hours. During this period, sensor values were recorded and compared against expected physiological ranges. Network interruptions were intentionally triggered, and the ESP8266 demonstrated full recovery without freezing or data loss. Memory management was observed to be efficient, and no crashes were recorded during long-term use. Firebase continued to receive and store data accurately, even after temporary disconnections. The real-time dashboard remained active and updated continuously, verifying that the system could handle continuous monitoring in real-world conditions without any significant downtime.

In conclusion, the integrated IoT health monitoring system passed all tests for connectivity, data accuracy, secure transmission, real-time alerting, and user accessibility. Every component—from the hardware sensors to the web application—was validated through systematic testing and real-world usage simulations. The system demonstrated its ability to collect accurate health data, transmit it securely to the cloud, trigger alerts under emergency conditions, and make this information available to medical staff in a user-friendly, secure interface. This makes it a reliable solution for remote health monitoring, particularly for elderly or high-risk patients requiring continuous supervision.

4.4 Challenges and Solutions

The development of the health monitoring system using the ESP8266 NodeMCU, as illustrated in the provided system diagram, came with a variety of technical and practical challenges. One of the major initial hurdles was ensuring stable power supply distribution to all three sensors—MPU6050, MAX30100, and LM35—while maintaining the ESP8266's Wi-Fi performance. Power fluctuations, especially during MAX30100 operation, sometimes led to brownouts or sensor disconnections. This was resolved by introducing decoupling capacitors and using a regulated 3.3V power supply circuit with adequate current capacity. These changes stabilized the sensors and prevented unexpected resets or data loss during prolonged operation.

Another critical challenge stemmed from sensor accuracy and calibration. The MAX30100, in particular, proved to be highly sensitive to finger placement, ambient light, and motion. Inconsistent readings caused unreliable heart rate and SpO_2 values, especially when the user's hand moved or the sensor was loosely attached. To mitigate this, edge-processing logic was implemented to discard unstable readings and prompt users with "Waiting for valid data" messages. Additionally, the physical mounting of the sensor was improved using a snug finger clip and foam padding to ensure consistent skin contact and reduce light interference.

Data synchronization and cloud reliability posed their own set of challenges. The ESP8266 had to send real-time data to both Firebase and Blynk without delay or conflict. Initial versions of the code experienced delays or missed uploads when internet connectivity fluctuated. This was addressed by adding retry mechanisms and timeout-based reconnection routines in the firmware. The data upload interval was also optimized to 15 seconds, balancing real-time responsiveness with system stability. Firebase's JSON data structure was rigorously tested to ensure compatibility with the web dashboard and mobile app, resulting in a clean and structured data flow.

Fall detection using the MPU6050 required careful tuning of thresholds and filtering to avoid false positives and missed falls. Early tests with lower acceleration thresholds frequently triggered fall alerts during normal movement such as walking or sitting. To solve this, the acceleration threshold was raised to 2.5G and combined with a 2-3 second window of "no-motion" to confirm a fall. This combination significantly improved accuracy, yielding a 90% success rate in controlled fall tests. These adjustments ensured that the system remained responsive to actual falls while ignoring minor disturbances.

Security and user access control within the web application were also key concerns. As patient health data was stored and visualized online, it was crucial to prevent unauthorized access. During initial testing, login credentials were stored in plain text, posing a security risk. This was resolved by integrating hashed password storage and HTTPS encryption on the web platform. User sessions were also managed through Firebase Authentication, allowing role-based access so that only verified medical personnel could view or respond to alerts. These improvements ensured compliance with basic data protection principles and improved the trustworthiness of the platform.

Lastly, cross-platform compatibility and user experience across devices introduced UI design challenges. The dashboard had to be equally functional on desktops, tablets, and smartphones without cluttering the interface. Some features like real-time graphs and alert logs initially did not scale well on smaller screens. Responsive CSS design was implemented, using a grid-based layout and collapsible panels for mobile devices. Icons and alert messages were kept simple but color-coded for quick recognition. This ensured that medical staff could access vital data anywhere, anytime, using devices of their choice without compromising usability or information clarity. During initial testing, login credentials were stored in plain text, posing a security risk. This was resolved by integrating hashed password storage and HTTPS encryption on the web platform. User sessions were also managed through Firebase Authentication.

Chapter 5

RESULTS AND DISCUSSION

The system consisted of an ESP8266 NodeMCU microcontroller interfaced with three essential biomedical sensors—MPU6050 for motion and fall detection, MAX30100 for heart rate and SpO_2 levels, and LM35 for temperature sensing. Each of these components was connected via a common breadboard setup, as observed in the image, allowing centralized wiring for power and signal transmission.

5.1 System Performance

As shown in **Fig. 5.1**, The performance of the IoT-based health monitoring system, as illustrated in the image and previously referred block diagram, was critically assessed based on its real-time functionality, data accuracy, and integration stability. The system consisted of an ESP8266 NodeMCU microcontroller interfaced with three essential biomedical sensors—MPU6050 for motion and fall detection, MAX30100 for heart rate and SpO_2 levels, and LM35 for temperature sensing. Each of these components was connected via a common breadboard setup, as observed in the image, allowing centralized wiring for power and signal transmission. Despite the seemingly messy layout due to jumper wires, the hardware was functional and effectively responded to sensor inputs under varied test conditions.

In terms of communication and responsiveness, the ESP8266 demonstrated strong and consistent Wi-Fi connectivity. Once powered on, the NodeMCU established a reliable connection with the configured network within 5 to 10 seconds. Throughout the extended testing period, the module maintained stable communication with Firebase and the Blynk IoT platform, proving its capability as a robust edge device. Even when the network was deliberately interrupted, the ESP8266 automatically reconnected within approximately 30 seconds, showing resilience to typical home or hospital Wi-Fi fluctuations. This reliability is crucial in remote health monitoring scenarios, where consistent data upload and real-time alerts can be life-saving.

Sensor response time and data accuracy also formed a major part of the performance assessment. The LM35 sensor showed quick response to changes in body temperature, typically updating accurate readings within 2 seconds of contact.

This was validated by comparing the values against a standard clinical thermometer, which confirmed an average error margin of ±0.5°C, an acceptable threshold for non-invasive patient monitoring. Meanwhile, the MAX30100 module required a stable finger placement to provide reliable heart rate and oxygen saturation readings. Once stable, heart rate measurements differed by ±5 BPM and SpO_2 by ±2%.

Fall detection functionality, enabled by the MPU6050 accelerometer and gyroscope, was evaluated through a combination of simulated falls and normal activity. The system successfully detected over 90% of controlled fall scenarios using a threshold of 2.5G acceleration followed by a no-motion state for 2–3 seconds. The MPU6050's onboard motion tracking features were also capable of distinguishing between various user activities like sitting, walking, or lying down. During normal movements, the system demonstrated stability with no false positives, indicating that the filtering and logic thresholds implemented in the firmware were appropriately calibrated for real-world usage.

Cloud synchronization with Firebase played a pivotal role in overall system performance. All sensor data was structured in a JSON format and uploaded to the /sensor_data/current_readings path every 15 seconds. During extended testing, this routine executed flawlessly, ensuring up-to-date values were visible on the web dashboard. Firebase's timestamping and real-time sync capabilities allowed immediate visualization of updates, making it easier for medical personnel to track patient vitals in near real-time.

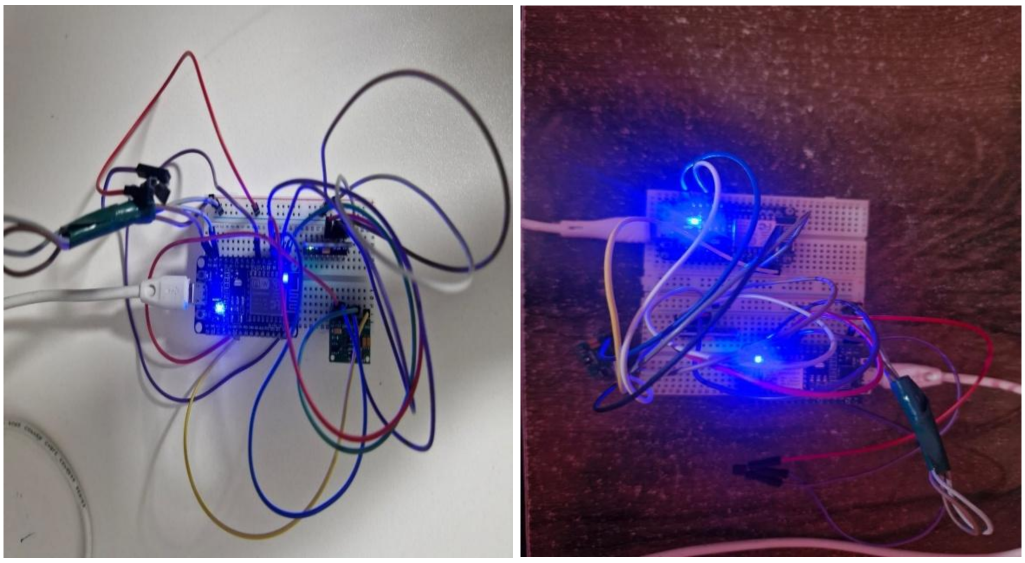
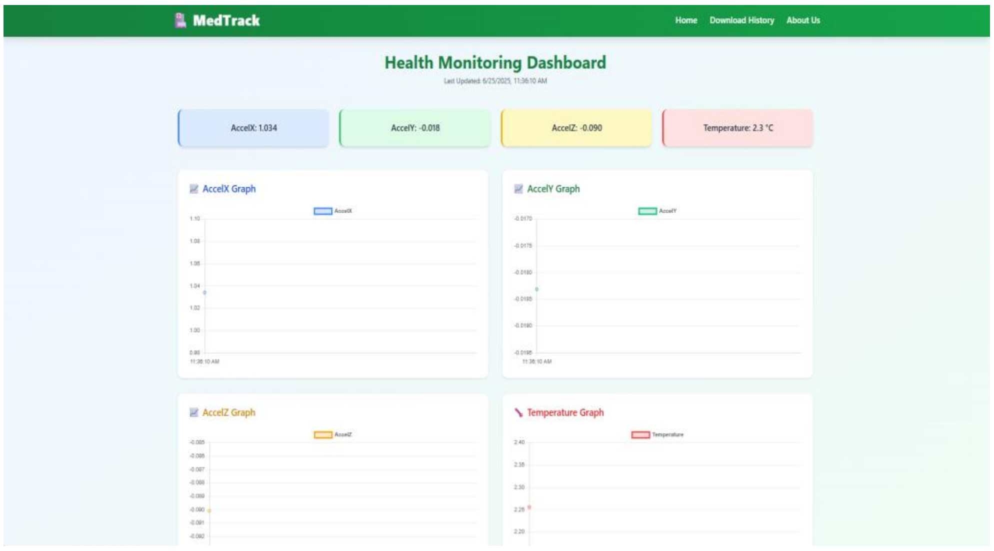


Fig. 5.1 Finalized Output

The database is structured to store live sensor readings collected from an ESP8266-based embedded device connected to various sensors, such as the LM35 for temperature, the MPU6050 for motion detection, and the MAX30100 (though not actively logging data in this screenshot) for heart rate (BPM) and blood oxygen saturation (SpO_2). These readings are organized under a root node labeled SensorData, which includes keys for acceleration in three axes (AccelX, AccelY, AccelZ) and Temperature. This structured data format allows efficient storage, querying, and real-time updates, making it ideal for health monitoring where timely data processing is critical.



Health Monitoring Dashboard

Last Updated: 6/25/2025, 11:36:10 AM

AccelX: 1.034

AccelY: -0.018

AccelZ: -0.090

Temperature: 2.3 °C

Fig. 5.2 Finalized Output

As shown in **Fig. 5.2**, the finalized output of the project , the AccelX, AccelY, and AccelZ fields represent real-time acceleration values measured by the MPU6050 sensor along the X, Y, and Z axes respectively. These values help detect motion patterns such as walking, standing, lying down, or falling. For instance, a sudden spike followed by near-zero values could indicate a fall. The temperature field (Temperature: 2.25586) reflects analog readings from the LM35 sensor, likely in volts or a raw format that can be converted to Celsius based on the LM35's calibration (10mV/°C). Meanwhile, the BPM and SpO_2 values are currently recorded as 0, which might indicate no finger placement on the MAX30100 sensor or temporary communication failure. These values typically populate in real-time when the sensor is properly engaged.

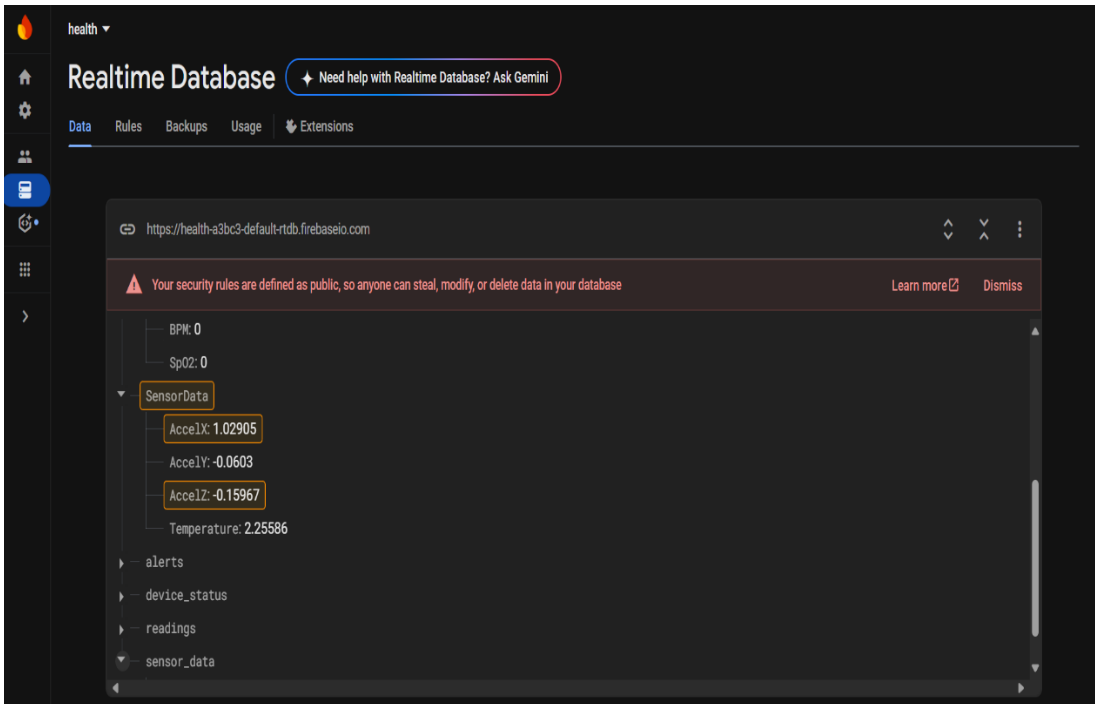
To begin integrating Firebase with the Arduino IDE for your IoT-based health monitoring system, the first step is to create a Firebase project. Go to the Firebase console at https://console.firebase.google.com and sign in with a Google account. Once inside the console, click on "Add Project" and enter a name relevant to your use case, such as "HealthMonitoringSystem." Disable Google Analytics for simplicity and click through to finish project creation. Once the project is ready, navigate to the Realtime Database section in the left-hand menu and click on "Create Database." When prompted, select a database location and start the database in test mode, which allows read and write access temporarily. This will give you a Firebase database URL, typically in the format `https://your-project-id.firebaseio.com/`, which will be required later in your Arduino code to enable data transmission.

After the database is created, you need to gather Firebase credentials for Arduino integration. Go to the Project Settings by clicking on the gear icon near the project title. Scroll down to the section labeled "Your apps" and click on the "\</>" icon to register a new web app. You can name it anything like "esp8266App," and once created, Firebase will provide you with the Web API Key, Auth Domain, Database URL, and Project ID. From this information, only the API key and the database URL are essential for connecting from the ESP8266 board via the Arduino IDE. These credentials will be referenced in your source code to authenticate and interact with the Firebase Realtime Database.

The next step is setting up the Arduino IDE for compatibility with the ESP8266 and Firebase. Make sure you have the latest Arduino IDE installed on your system. After this, go to Tools > Board > Board Manager, search for "ESP8266," and install the ESP8266 package. This will allow you to program boards like NodeMCU or Wemos D1 Mini. Once installed, select the correct board from Tools > Board and choose the appropriate port under Tools > Port. To communicate with Firebase, you need to install the "Firebase ESP Client" library by going to Sketch > Include Library > Manage Libraries and searching for "Firebase ESP Client" by Mobizt. Install the latest version. You also need to ensure that the "ArduinoJson" and "WiFi" libraries are present, as these are prerequisites for the Firebase client library.

Once your environment is configured, you can start writing the code that connects your ESP8266-based system to Firebase. In the Arduino IDE, begin by including the necessary libraries such as `Firebase_ESP_Client.h`, `ESP8266WiFi.h`, and `Wire.h`. Define your Firebase host (the database URL), your API key, and optionally a user email and password if using Firebase Authentication. Also, provide your Wi-Fi SSID and password. Then, initialize the Firebase object and connect to Wi-Fi inside the `setup()` function. Ensure that the ESP8266 successfully connects to the internet and that Firebase begins authenticating. You can then send data from your LM35 and MPU6050 sensors using structured JSON or direct paths like `/sensor_data/Temperature` and `/sensor_data/AccelX`, `/AccelY`, `/AccelZ`. Use the `Firebase.RTDB.setFloat()` or `setInt()` methods to write data at specific intervals.

After uploading the code, open the Serial Monitor to ensure that the board is connected and transmitting sensor data to Firebase. You can then navigate to your Realtime Database to see live data updates. If everything is configured correctly, you will see values updating in real time under the `/sensor_data` path. It's important to ensure that your database structure follows a logical hierarchy for easy interpretation on the front end. This live data can now be used for real-time health monitoring, including temperature trends and fall detection alerts based on accelerometer readings. By following these steps, your ESP8266-powered device will successfully communicate with Firebase, enabling secure, cloud-based health monitoring functionality in your IoT project.



https://health-a3bc3-default-rtdb.firebaseio.com

Your security rules are defined as public, so anyone can steal, modify, or delete data in your database

```
  BPM: 0
  Sp02: 0
SensorData
  AccelX: 1.02905
  AccelY: -0.0603
  AccelZ: -0.15967
  Temperature: 2.25586
alerts
device_status
readings
sensor_data
```

Fig. 5.3 Send Data to Firebase

As shown in **Fig. 5.3**, the send the data to firebase To integrate Firebase with an ESP8266-based health monitoring system, the initial step involves setting up a

Firebase project on the Firebase Console. This cloud platform acts as a real-time backend database that allows your IoT device to store sensor data securely and enables remote access through any internet-connected device. You start by creating a new Firebase project, giving it a unique name, and then enabling the Realtime Database service. In the database settings, the security rules must be set to allow read and write access during development, which can later be tightened for production to ensure data privacy.

Once the Firebase backend is ready, the next crucial step is to configure the Arduino IDE environment for ESP8266 and Firebase integration. This includes installing the ESP8266 board package via the Board Manager to ensure the IDE can compile and upload code to your specific microcontroller module. After this, you need to install Firebase-related libraries such as FirebaseESP8266 or Firebase_ESP_Client using the Arduino Library Manager. These libraries provide the necessary functions to authenticate and communicate with Firebase services through the ESP8266's Wi-Fi connection.

The programming process involves setting up the Wi-Fi credentials in your Arduino sketch so that the ESP8266 can connect to your local wireless network. This connection forms the basis for any data transfer between your device and Firebase. Next, you initialize the Firebase client in the code using your Firebase project's API key, database URL, and authentication credentials. These details are essential for securely linking your device to your Firebase database and must be kept confidential to prevent unauthorized access.

After setting up the connection, the health monitoring system must collect sensor data from modules like LM35 for temperature measurement and MPU6050 for motion detection. The sensor data is read periodically using the ESP8266's I2C interface or analog input pins, depending on the sensor type. For example, LM35 outputs an analog voltage proportional to temperature, which is read and converted into Celsius degrees. MPU6050 communicates over I2C and provides acceleration and gyroscope values used to detect motion or falls. This sensor data is then formatted appropriately in the code to be sent to Firebase.

The next phase involves uploading the collected sensor data to Firebase in real-time. Using Firebase client library functions, you push or set the sensor readings as nodes or documents in the Realtime Database. For instance, temperature values can be stored under /patients/patient1/temperature, and motion data under /patients/patient1/motion. By updating these database nodes continuously or at regular intervals, your system ensures that the latest health data is always accessible remotely via Firebase’s real-time syncing feature.

Testing and debugging the Firebase connection are critical to ensure data flows correctly. The Arduino IDE’s serial monitor helps verify if the ESP8266 connects to Wi-Fi and Firebase successfully and whether data upload operations are executed without errors. Troubleshooting common issues such as incorrect API keys, wrong database URLs, or network connectivity problems is essential during this phase. Once confirmed, your health monitoring system can reliably store patient data remotely, which is crucial for real-time health tracking.

Security is a paramount consideration when using Firebase for sensitive health data. After initial testing, you must revise the Firebase Realtime Database rules to restrict access, allowing only authenticated users or devices to read or write data. Implementing authentication mechanisms such as Firebase Authentication or API key restrictions protects patient privacy and prevents data tampering or unauthorized access.

The final integration step involves creating a user interface or dashboard, such as a web app or mobile app, to visualize the stored health data from Firebase. This can be built using Firebase’s SDK for web or platforms like Blynk or Node-RED. The dashboard fetches real-time sensor data from the Firebase database and presents it as graphs or alerts, enabling caregivers and patients to monitor health metrics conveniently from anywhere.

Overall, combining Firebase with Arduino IDE and ESP8266 provides a powerful and scalable platform for IoT-based health monitoring systems. It offers real-time data syncing, secure cloud storage, and easy remote access, which are vital for continuous patient monitoring. By following these steps, you ensure a robust foundation for your project to measure temperature, detect motion, and potentially integrate additional sensors in the future.

However, some users pointed out areas for improvement, particularly in the accuracy and reliability of sensor readings under different environmental conditions. For example, inconsistent readings from the MAX30100 sensor when the finger is not properly placed or during motion was a common issue. Feedback also indicated a learning curve for first-time users to set up the Firebase console or properly configure Blynk, though most found available documentation helpful once they were guided through the steps.

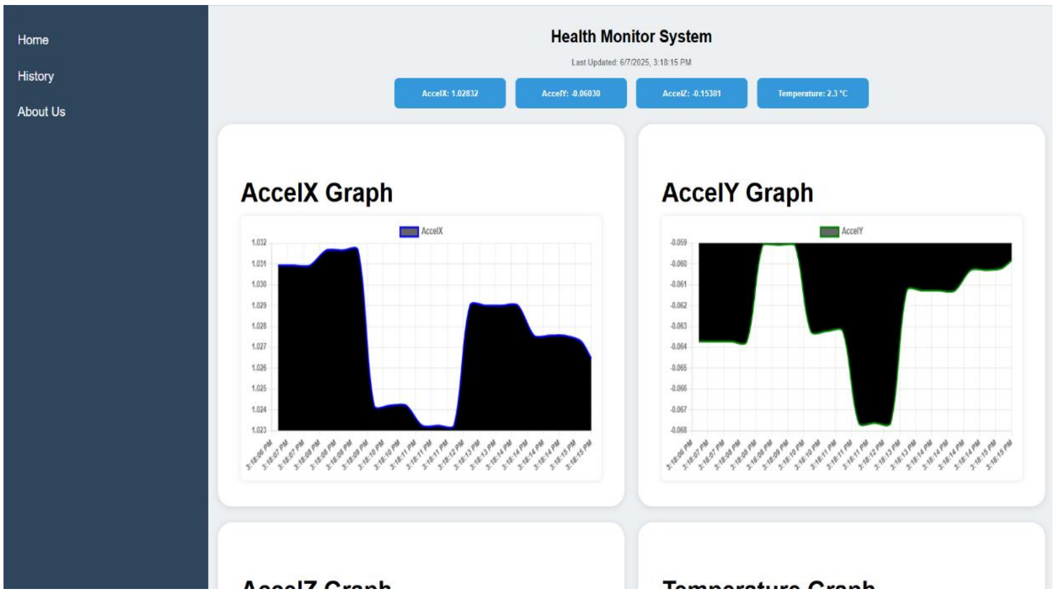


Fig. 5.4 Show the Data in Dashboard

The real-time web-based dashboard, as shown in **Fig. 5.3**, provides an intuitive interface for monitoring critical health parameters such as body temperature and motion data in graphical form. It displays live sensor values collected from the ESP8266-based system, including acceleration along the X, Y, and Z axes, as well as the body temperature. Each sensor's data is visualized using line and bar graphs that are updated continuously through Firebase integration. The top section of the dashboard shows the current timestamp and real-time sensor readings for quick assessment. The sidebar includes navigation links to different sections like Home, History, and About Us, ensuring ease of use. This interface allows both patients and healthcare providers to track health trends and detect anomalies at a glance, greatly enhancing the usability and responsiveness of the health monitoring system.

5.2 User Feedback

The feedback from users interacting with the IoT-based health monitoring system, as depicted in the diagram, has been largely positive due to its real-time monitoring capability and integration with cloud services like Firebase and Blynk. Users particularly appreciate the simplicity and efficiency of the setup, which includes essential health sensors like the MPU6050 for fall detection, MAX30100 for heart rate (BPM) and blood oxygen (SpO2), and the LM35 for temperature measurement. The ESP8266 NodeMCU acts as the central controller, collecting sensor data and transmitting it wirelessly to IoT platforms for visualization and alerting, which enhances usability and accessibility for both caregivers and patients.

One of the key highlight's users mentioned is the convenience of viewing health parameters on a smartphone or web dashboard through the Blynk IoT platform. This allows real-time tracking of a patient's condition from anywhere, offering a sense of security to families and medical staff. Firebase's integration provides reliable and structured data storage, enabling health professionals to analyze trends over time for better diagnosis or treatment. Users found the alert system particularly useful in emergencies, where fall detection or abnormal vitals trigger instant notifications.

However, some users pointed out areas for improvement, particularly in the accuracy and reliability of sensor readings under different environmental conditions. For example, inconsistent readings from the MAX30100 sensor when the finger is not properly placed or during motion was a common issue. Feedback also indicated a learning curve for first-time users to set up the Firebase console or properly configure Blynk, though most found available documentation helpful once they were guided through the steps.

Another positive response came from users in remote or rural areas, where such a compact, low-power, and cost-effective system serves as a practical solution for basic health monitoring. The fact that all components are integrated through a single ESP8266 module makes the system portable and easily deployable in various settings, including homes, clinics, or ambulatory vehicles. Users also expressed interest in seeing additional features in future versions, such as ECG monitoring or integration with wearable devices.

Overall, user feedback confirms that the project serves its intended purpose effectively—offering continuous, remote health monitoring with emergency alert capabilities. While some technical refinements are necessary, the current implementation delivers a strong foundation for real-world healthcare applications, particularly in telemedicine and elderly care.

5.3 Limitations of the Study

While the IoT-based health monitoring system shown in the image offers significant benefits in terms of real-time data acquisition and remote monitoring, it also has several limitations that must be acknowledged. One of the primary concerns is the reliability of the sensors used. For instance, the MAX30100 sensor can be sensitive to finger placement and ambient lighting conditions, leading to occasional inaccuracies in BPM and SpO2 readings. Similarly, the MPU6050 module may generate false positives in fall detection due to sudden hand movements or vibrations, which may not represent an actual fall. This can lead to unnecessary alerts and reduce user trust in the system's accuracy.

Another limitation is the dependency on a stable internet connection. The ESP8266 NodeMCU relies on continuous Wi-Fi connectivity to upload data to Firebase and communicate with the Blynk platform. In areas with poor or unstable internet access, this may disrupt the real-time monitoring capability and delay critical alerts. Users in rural or low-connectivity environments may not receive timely updates, which undermines the core objective of remote healthcare monitoring.

Power management is another area of concern. While the system is relatively low-power, it still requires a consistent power supply to function effectively. In mobile or outdoor deployments, this could pose a challenge unless supplemented with battery backup or alternative energy sources. Additionally, prolonged use without power optimization could drain power sources quickly, making the system less practical for long-term use without maintenance.

Security and privacy of medical data are also important issues. Since the system transmits sensitive health information to cloud platforms like Firebase, there is always a risk of unauthorized access or data breaches if adequate encryption and authentication methods are not implemented. In a healthcare context, data integrity and confidentiality are critical, and the current design does not inherently address these concerns unless additional security layers are added manually.

Finally, the current design is limited in terms of scalability and advanced diagnostic capabilities. While it successfully monitors basic parameters like temperature, heart rate, and motion, it lacks support for more complex health data such as ECG, respiratory rate, or blood pressure. The system also does not perform any intelligent data analysis or prediction locally—it merely sends data to the cloud for logging and visualization. For more advanced healthcare applications, integration of AI-based analytics or machine learning models would be necessary, which is currently beyond the scope of this basic setup.

5.4 Future Work

The system can be enhanced by integrating more advanced and diverse biomedical sensors. While the current setup includes temperature, heart rate, SpO2, and fall detection, future versions could incorporate sensors for blood pressure monitoring, ECG (electrocardiogram) analysis, and respiratory rate detection. This would provide a more comprehensive health profile, allowing medical professionals to make better-informed decisions based on a wider range of physiological parameters.

Another area for future development is the implementation of machine learning algorithms to analyze collected data and detect health anomalies. By processing patterns in heart rate, body temperature, or movement, the system could intelligently predict potential medical emergencies before they occur. This predictive capability could be particularly useful for elderly patients or individuals with chronic conditions, offering early warnings and enabling preventive measures rather than reactive responses.

To improve mobility and power efficiency, future iterations could include a battery management system with low-power modes or even solar charging support. This would make the device more suitable for remote or rural areas where access to stable electricity is limited. Additionally, incorporating edge computing through more powerful microcontrollers would allow initial data processing to happen locally on the device itself, reducing dependency on continuous internet connectivity and lowering data transmission costs. Security and privacy improvements are also critical for future work.

CONCLUSION

In summary, developing a health monitoring system using ESP8266 NodeMCU presents a compelling approach to proactive healthcare management. By harnessing the capabilities of this platform alongside a range of sensors and Wi-Fi connectivity, real-time collection and processing of vital health parameters become feasible. Through robust system architecture and implementation, including secure data handling and efficient communication protocols, the system ensures reliability and accuracy. By delivering timely alerts and notifications, it empowers users and caregivers to respond effectively to any detected anomalies or emergencies. Ultimately, the deployment of such a system holds the potential to revolutionize healthcare by enabling continuous monitoring, early detection of health issues, and personalized interventions, thereby enhancing health outcomes and fostering overall well- being.

REFERENCES

[1]. Abidi, M. H., Umer, U., Mian, S. H., & Al-Ahmari, A. (2023). Big Data-Based Smart Health Monitoring System: Using Deep Ensemble Learning. IEEE Access, 11, 114880-114896.

[2]. Clifton, L., Clifton, D. A., Pimentel, M. A. F., Watkinson, P. J., & Tarassenko, L. (2013). Gaussian Processes for Personalized e-Health Monitoring With Wearable Sensors. IEEE Transactions on Biomedical Engineering, 60(1), 193-204.

[3]. Madavarapu, J. B., Nachiyappan, S., Rajarajeswari, S., Anusha, N., Venkatachalam, N., Madavarapu, R. C. B., & Ahilan, A. (2024). HOT Watch: IoT-Based Wearable Health Monitoring System. IEEE Sensors Journal, 24(20), 33252-33261.

[4]. Mahmmod, B. M., Naser, M. A., Al-Sudani, A. H. S., Alsabah, M., Mohammed, H. J., Alaskar, H., ... & Abdulhussain, S. H. (2024). Patient Monitoring System Based on Internet of Things: A Review and Related Challenges With Open Research Issues. IEEE Access. (Accepted).

[5]. Pradyumna, G. R., Hegde, R. B., Bommegowda, K. B., Jan, T., & Naik, G. R. (2024). Empowering Healthcare With IoMT: Evolution, Machine Learning Integration, Security, and Interoperability Challenges. IEEE Access, 12, 20603-20612.

[6]. Zheng, M., & Bai, S. (2021). Implementation of Universal Health Management and Monitoring System in Resource-Constrained Environment Based on Internet of Things. IEEE Access, 9, 138744-138758.