

CHAPTER 1

INTRODUCTION

As we all are aware, heart attack-related death and disability rates are increasing in India every day. The Registrar General of India reported that, in 2001–2003, cardiovascular diseases were the cause of 17% of all deaths and 26% of adult deaths. In the period of 2010–2013, this figure reached 23% of all deaths and 32% of adult deaths. The government allocates a huge amount of money in every fiscal year for the health budget, and all those procedures are performed at lowered costs. This system makes it easy to diagnose and treat people who have heart diseases. With this technique, the doctor is able to remote-diagnose from where the patients are—whether that be the patient's home—through the cloud platform. Patients can view their medical records using this as well. Through this cloud service. Although there are so many different kinds of ECG recorders currently available on the market that are offered by reputable firms, there are now relatively very few gadgets which can capture the ECG signals and send these to a far distant database server on the cloud. In the present study, we have presented a system wherein a sensor records the patient's ECG signals and stores it on a database server. These signals can be recorded and recovered for study at a later time, or they can be examined remotely by a physician. Since a traditional ECG monitor measures the electrical activity of the heart only for short periods of time, it's likely that no heart problems exist at that moment. Therefore, what is required is a real-time system that can assess heart rate at any time.

The advancement of biomedical engineering is responsible for the improvement of medical diagnosis, treatment, and monitoring. The innovative concept behind Health Line is to provide high-quality healthcare to everyone. The concept of a cable-free biomedical monitoring system is what motivates the notion. On-body sensors track vital signs like blood pressure, ECG, temperature, and heart rate and send the information to the doctor via a wireless network. Regular health monitoring, often known as preventative care, enables people to identify and address health issues before they have an impact.

LITERATURE REVIEW

2.1 General Introduction

The discharged patient can use the IoT-based remote health monitoring system, which provides smart technology with the capacity for tracking and regulating health conditions beyond the hospital door. Wearable devices and sensors use real-time acquisition of data associated with heart rates, blood pressures, and oxygen saturation levels, after which such information is forwarded into cloud platforms through which doctors will be able to monitor the health status of these patients. Abnormal reading alerts are generated to ensure swift medical response. Personalized health information is given to patients, and this motivates patients to be proactive in self-care. It optimizes the use of healthcare resources, enhances patient results, and reduces hospital readmissions. It connects doctors and patients through mobile applications. Data encryption serves to ensure security and confidentiality. Considering all this, it makes available to patients and healthcare providers a cost-effective and efficient healthcare resource.

Random Forest Algorithms

Numerous industries and fields, including agriculture, transportation, electricity, and healthcare, have benefited from the attention-grabbing tree-based ensemble learning algorithms. These techniques have been used to forecast outcomes across a range of healthcare settings. When designing a healthcare system, clinical outcomes, healthcare expenditures, and healthcare efficiency are considered. Because these algorithms automatically handle interactions, they are also accurate predictors, even when a large number of variables are present. Random Forests are a type of ensemble approach based on trees that is commonly used for classification and regression. Numerous recent studies have yielded reassuring empirical and theoretical findings. Additionally, RFs are gaining traction in the healthcare industry, where they can be used for various purposes. A large number of decision trees are generated by RFs. These trees are constructed by substituting variables from a randomly chosen subset of the original dataset for variables from the randomly chosen subset. RFs are capable of dealing with both categorical and numerical

variables in prediction situations. One of the characteristics of RFs is their built-in cross-validation capability, which enables the independent variables to be ranked according to their association with the outcome variable, from most effective to least effective. This simplifies the process of extracting functions from multisource data analysis. Breiman's presentation provides an in-depth overview of radio frequency technology. While radiofrequency devices have the potential to automate a wide variety of operations on large and complex datasets, their utility in the context of particular safety culture has yet to be thoroughly investigated and evaluated. Researchers intend to build on the result of this research by combining an entirely new application field with a specialized

2.2 Literature Review

1. Augustus E. Ibhaze, MNSE, Francis E. Idachaba, “Health Monitoring System for the Aged” 2016 IEEE, International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech)

These wearable and mobile technologies are based on monitoring the key vital health parameters, including heart rate, blood pressure, and body temperature. This gadget collects the data in real-time and transfers wirelessly to a central server for analysis. In case these abnormal readings take place, this system is expected to inform immediately the caretakers or the medical specialists so appropriate interventions are undertaken on time. This would improve the quality of life among seniors through continuous monitoring for health challenges, especially in those elderly citizens who are solitary or very far from any medical facility. It is cost-effective and efficient in healthcare delivery as it minimizes rehospitalization and ensures pro-active management of chronic diseases. The article also emphasizes the importance of integrating strong communication protocols and secure data transmission to ensure the dependability and privacy of the monitoring system.

2. Lakmini P. Malasinghe et al,(2017) “Remote patient monitoring: a comprehensive study”, Springer

Technologies and methodologies for monitoring patient care outside the traditional clinical setup will be based on wearable sensors, mobile devices, and cloud-based platforms. The devices might continuously monitor the status of the patients, collect, and transmit real-time physiological data pertaining to heart rate, blood pressure, glucose levels, and respiratory parameters. The authors outline many benefits of RPM, including advantages

for better patient outcomes, reduced readmissions to hospitals, and cost-effectiveness. Other topics covered in the report are interoperability issues, data security concerns, and the patient's ability to use the technology. The report illustrates the vast potential of RPM in revolutionizing postoperative care, chronic disease management, and care for the elderly by looking into a variety of case studies and applications. Therefore, the authors do support the robustness of standards so that the widely adopted and effectively functioning RPM system will have better safety measures, besides a simplified user interface.

3. Abhilasha Ingole, Shrikant Ambatkar, Sandeep Kakde, "Implementation of Health-care Monitoring System using Raspberry Pi", IEEE ICCSP 2015 conference., 2015 IEEE.

A central processing unit of this system is obtained from a Raspberry Pi microcontroller and integrated with multiple biomedical sensors for monitoring vital parameters, including heart rate, body temperature, and blood pressure. For easy understanding, the real-time data collected by those sensors are processed and depicted on graphical user interface. One of the key features of the system is the fact that it transmits data over the internet and allows a caretaker and medical expert to monitor the patient over a distance. Anomalous readings generate alerts for timely medical interventions. According to the authors, the cost of the system makes it suitable for widespread utilization, especially in rural or resource-constrained environments. This research demonstrates how small, strong devices like Raspberry Pi can form the basis for innovative medical applications that reduce dependency on traditional monitoring systems and improve patient care. Some of the possible improvements would be the increasing scalability of the system and more sensors added to it.

4. Skraba, Andrej, et al. "Prototype of Group Heart Rate Monitoring with NODEMCU ESP8266." 6th Mediterranean Conference on Embedded Computing (MECO)

It uses a low-cost and Wi-Fi-enabled NodeMCU ESP8266 microcontroller to collect data on heart rates using wearable pulse sensors. It then transmits these data wirelessly to the central server that processes these and makes them viewable through a friendly interface for real-time monitoring of a group. The authors explain the potential uses of the system in fitness centers, sports training, and monitoring the health status of groups of people in a public setting like a workplace or school. Hardware is low cost and dependent on open-

source software that makes it available and scalable for various users. The system has alert capabilities regarding abnormal heart rate readings that ensure the taking of swift action in cases of a probable health problem. The study concludes by discussing some of the challenges such as accuracy of data, signal interference, and further development to support more robust functionalities and larger user groups. This prototype demonstrates the practicality of IoT-based solutions for real-time, group health monitoring.

5. D. Hasan and A. Ismaeel, “Designing ECG Monitoring Healthcare System Based on Internet of Things Blynk Application,” J. Appl.Sci. Technol. Trends

The integration of IoT technology in healthcare has transformed the way patients are monitored continuously and in real-time, monitoring critical health parameters. Hasan and Ismaeel (2020) proposed an ECG monitoring system based on the Blynk application that was effective in remote health tracking. The system collects and transmits ECG data efficiently and allows timely access to healthcare professionals. The authors successfully demonstrated how their system can add value to accessible health services with minimal in-person consultation requirements using the IoT frameworks. The Blynk application is quite user-friendly; it can allow a patient and also the healthcare professional to view live data and develop alerts in a straightforward manner.

6. F. Bamarouf, C. Crandell, S. Tsuyuki, J. Sanchez, and Y. Lu, “Cloud-based real- time heart monitoring and ECG signal processing,” in 2016 IEEE SENSORS, Orlando, FL, USA, Oct. 2016

The advent of cloud computing has revolutionized healthcare, especially in the area of real-time cardiac monitoring. Bamarouf et al. (2016) proposed a cloud-based system for real-time heart monitoring and ECG signal processing that could make cardiac care more efficient and accessible. With the cloud infrastructure, it can collect, process, and analyze data in real time and detect and respond to critical cardiac events in time. The study shows that the cloud framework is excellent in the processing of massive amounts of ECG data with high accuracy and speed, thus reducing dependence on local hardware and making it more portable and cost-effective. Moreover, it is appropriate for use in applications individually and at large scales. Besides the above, research emphasized the systems' ability to address traditional limitations in cardiac monitoring, such as proximity to a healthcare facility, constraints in data analysis in real time. In addition, the study

mentioned that robust security and reliability would be required in the data if it were implemented effectively and safely. This work helps to gain important insights on the integration of cloud technology into healthcare and offers a potential avenue for advancing patient outcomes by making early cardiac abnormalities detection feasible

2.3 Summary

With the integration of IoT and advanced technologies, remote health monitoring systems are developed for patients discharged from the hospitals. Several studies show the utilization of wearable sensors, mobile devices, and microcontrollers such as Raspberry Pi and NodeMCU ESP8266 to collect real-time physiological data like heart rate, blood pressure, and body temperature, and transmit it to the centralized servers through wireless communication. These systems utilize cloud platforms for the efficient processing, storage, and analysis of data to enable the remote monitoring of patients by healthcare providers. Its key features include user-friendly interfaces, real-time alerts for abnormal readings, and scalability for individual or group monitoring. The systems aim to enhance the accessibility of health care, reduce readmissions to the hospital, and promote cost-effective care, particularly in resource-poor settings. Addressing the challenges of data security, interoperability, and signal accuracy with a focus on robust protocols and new functionalities, solutions based on IoT show solid potential towards active patient management and betterment in post-discharge healthcare.

PROBLEM STATEMENT AND OBJECTIVE

3.1 Problem Statement

Current post-discharge care lacks continuous monitoring and timely intervention, raising healthcare costs and readmission rates.

3.2 Objective

Developing a cost-effective remote health monitoring system for discharged patients, capable of real-time data collection, analysis, and timely intervention to reduce hospital readmissions and improve patient outcomes.

METHODOLOGY

4.1 Working flow chart

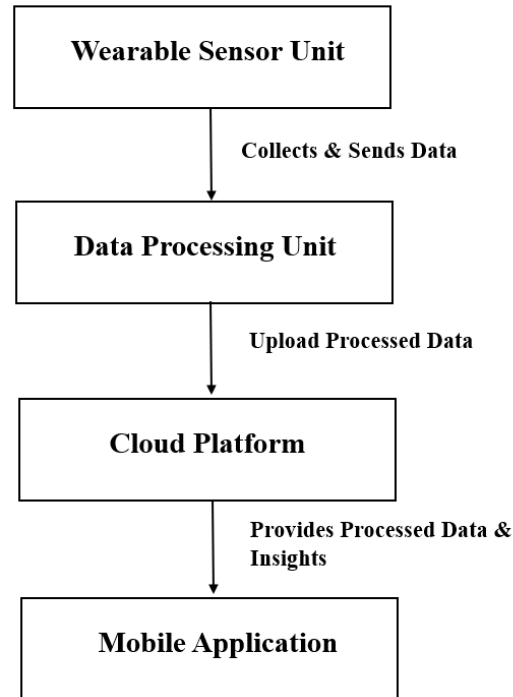


Figure 4.1 Working Flow Chart

Wearable Sensor Unit

The Wearable Sensor Unit serves as the system's primary component, tasked with gathering raw physiological data from the user's body. This unit typically incorporates various sensors, such as a Galvanic Skin Response (GSR) sensor, a body temperature sensor, and a heart rate sensor, depending on the specific application. These sensors continuously monitor and record the user's physiological parameters. To facilitate seamless data transmission for further processing, the unit is equipped with connectivity options like Bluetooth or Wi-Fi.

Data Processing Unit

The Data Processing Unit acts as an intermediary between the Wearable Sensor Unit and the cloud platform. Its primary role is to filter, analyse, and preprocess the raw data received from sensors, removing noise and extracting valuable information, such as detecting irregularities or identifying patterns. This preprocessing ensures efficient data transmission and reduces the

burden on the cloud platform. In many cases, microcontrollers or processors integrated with embedded algorithms are utilized for real-time analysis. The processed data is then transmitted to the cloud platform for advanced analytics and storage.

Cloud Platform

The Cloud Platform functions as a centralized system for data storage, analysis, and management. It securely stores the data uploaded by the Data Processing Unit and performs in-depth analyses to generate actionable insights. These insights may include health trends, alerts for anomalies, or tailored recommendations. Additionally, the cloud platform bridges the wearable devices with the mobile application, facilitating real-time communication and data access for users.

Mobile Application

The Mobile Application serves as the user-facing interface for interacting with the system. It retrieves processed data and insights from the cloud platform and displays them in a visually engaging and easy-to-understand format, such as graphs, alerts, or notifications. Users can configure device settings, request additional data, or trigger specific actions, like sending alerts during emergencies. The application provides real-time monitoring and feedback, empowering users to manage their health or safety proactively.

4.2 Block Diagram

The proposed system integrates advanced sensor technologies and IoT frameworks to provide an efficient solution for health monitoring. It employs a variety of sensors tailored to the application, such as temperature sensors to measure body heat and heart rate sensors to track cardiovascular activity. These sensors continuously monitor physiological parameters, ensuring accurate and real-time data acquisition. An Arduino Uno serves as the central processing unit, collecting raw data from the sensors, processing it, and displaying the readings on an LCD screen for immediate reference. Simultaneously, the system leverages a Wi-Fi module to transmit the processed data to a cloud-based platform for advanced analytics and storage.

The IoT cloud platform acts as a bridge between the hardware and software components, enabling healthcare professionals to access patient data remotely through web or mobile applications. This ensures real-time monitoring and allows for quick decision-making in case

of abnormalities. Alerts are generated automatically and sent to the user's smartphone when sensor readings fall outside the set threshold, ensuring timely interventions.

The system incorporates components such as the LM35 temperature sensor, which is highly precise and offers linear output across a wide temperature range, making it ideal for monitoring body temperature. For ECG monitoring, the AD8232 module provides robust performance by amplifying and filtering weak bio-signals while reducing noise and motion artifacts. Its excellent Common Mode Rejection Ratio ensures the accuracy of the ECG data.

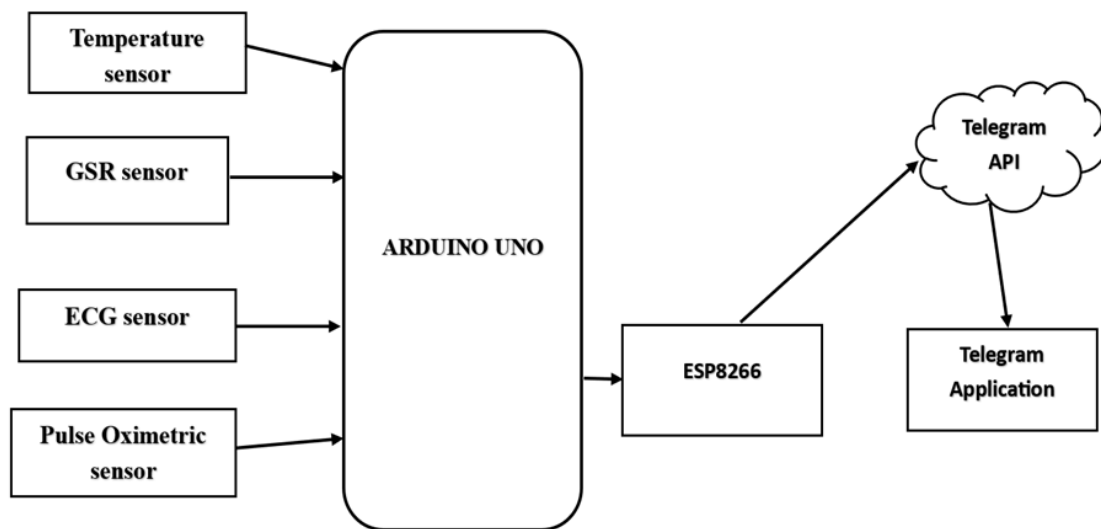


Figure 4.2 Block Diagram of IOT-Based Remote Health Monitoring System for Discharged Patient

By integrating these components with IoT capabilities, the system enhances accessibility, allowing users to monitor their health or that of patients effectively from anywhere. This approach is particularly useful for managing chronic conditions, elderly care, and remote healthcare applications. Additionally, the use of an intuitive interface and automated notifications makes the system user-friendly and suitable for a broad range of applications, promoting proactive healthcare and reducing dependency on traditional monitoring methods.

SOFTWARE AND HARDWARE REQUIREMENTS

5.1 Software Requirements

The Arduino Integrated Development Environment, or IDE, is a free, open-source platform that makes it easier for a person to write and upload code onto Arduino boards. It supports all the three common operating systems, such as Windows, macOS, and Linux. Being developed in Java, it's based on Processing and other open-source technologies, which makes this a very versatile environment for Arduino-compatible hardware.

The Arduino IDE has an intuitive interface that offers a text editor to write and edit code, a message area to give back feedback or error messages, and a console that outputs text as well as diagnostic messages. There is a toolbar through which one can perform common activities such as verification, upload, and saving of the code. It displays the board type that is connected along with the serial port for configuration.

The code written in the IDE is called "sketches" and is saved with a file extension of .ino. The editor contains standard text manipulation tools available, such as copy, paste, and search-and-replace functions. The IDE offers programming support in C and C++ with special rules to structure its code. Its library also contains a version of software borrowed from the Wiring project, which assists users in many input/output tasks.

Each of these sketches should have at least two procedures: an initialization function and a main program loop function. The pre-defined main function is used to compile and link the sketches into an executable program. The steps of compiling are carried out by the GNU toolchain, which is self-contained within the IDE. The program finally is converted into a hexadecimal text file and uploaded to the Arduino board using a loader program. The IDE does this by using avrdude to flash your code onto officially licensed Arduino boards.

The Arduino platform has seen the evolution of the IDE from the Processing framework to embracing the Visual Studio Code-based Eclipse Theia framework since version 2.0. This has modernized it and increased its support for third-party microcontrollers as well. As Arduino gained popularity, other developers started developing open-source tools and compilers to ease support for more hardware that might not have been officially supported in the Arduino sphere, thus making it a versatile platform for any range of applications.

Microsoft Excel

Microsoft Excel is one of the most widely used spreadsheet software globally, offering robust tools for data management, analysis, and visualization. Introduced by Microsoft, it has grown into an essential application for professionals, students, and organizations. Excel's primary function is to store and organize data in rows and columns, making it easily accessible and analysable. Its formula capabilities allow users to automate calculations ranging from basic arithmetic to complex financial modelling, such as using SUM, AVERAGE, VLOOKUP, or IF functions. Beyond calculations, Excel excels at creating visually appealing charts, such as bar graphs, line graphs, scatter plots, and pie charts, which help users understand data patterns. Pivot Tables are another standout feature, enabling users to summarize, filter, and manipulate large datasets effortlessly. For advanced users, the inclusion of Macros and VBA (Visual Basic for Applications) offers automation of repetitive tasks, significantly improving efficiency. Excel also supports real-time collaboration through Microsoft 365, allowing multiple users to work on the same spreadsheet simultaneously. It includes features for data validation to prevent errors during data entry, as well as password-protected sheets to secure sensitive information. Conditional formatting further enhances data readability by allowing users to highlight critical values based on predefined conditions. Excel's ability to handle external data integration is also notable, enabling users to import/export data from databases, APIs, and other files, such as CSV or XML. The software supports data analysis tools like Solver, Goal Seek, and the Analysis ToolPak for statistical or optimization tasks. Its seamless integration with other Microsoft Office tools, including Word and PowerPoint, streamlines workflows. Excel is available across multiple platforms, including Windows, macOS, iOS, and Android, ensuring accessibility. In professional environments, it is used for financial modelling, budgeting, project planning, and performance tracking, while in education, it is a preferred tool for teaching data skills. Advanced charting tools, combined with slicers and timeline filters, make dashboard creation intuitive and interactive. Furthermore, Excel's extensive online community and resources provide support and templates for almost every imaginable use case. Updates to Excel have consistently introduced AI-powered features, such as Ideas, which suggest insights from data, and XLOOKUP, a modernized replacement for VLOOKUP and HLOOKUP. With support for add-ins and extensions, users can customize Excel to fit their needs, extending its capabilities beyond traditional spreadsheet applications. Its adaptability to different industries, from healthcare to engineering, demonstrates its versatility. Businesses rely on Excel for inventory tracking, sales forecasting, and customer data analysis. The combination of ease of

use and advanced functionalities ensures that Excel remains a cornerstone of productivity tools. Moreover, the capacity to work with big data has improved, with Excel now offering features to handle millions of rows without crashing. Training programs and certifications are widely available for mastering Excel, making it a valuable skill in the job market. The built-in Power Query tool simplifies data cleaning and transformation tasks, while Power Pivot enhances data modelling and reporting capabilities. Excel's use cases are virtually endless, ranging from creating academic grade sheets to monitoring global economic trends. Its compatibility with cloud storage ensures files are safe, accessible, and shareable, no matter the device. Overall, Excel is a sophisticated, user-friendly tool that caters to beginners and advanced users alike, empowering them to make informed decisions based on well-organized and analyzed data.

Microsoft Visual Studio

Microsoft Visual Studio is a robust integrated development environment (IDE) designed for developers to create, test, and deploy applications across multiple platforms. Visual Studio supports programming languages like C#, C++, Python, JavaScript, and more, making it a versatile tool for software development. Its intuitive interface includes features like IntelliSense, which provides smart code suggestions, and debugging tools that help developers identify and fix issues efficiently.

The IDE supports project templates, enabling developers to start projects quickly with predefined configurations for web, desktop, and mobile apps. Visual Studio also integrates seamlessly with Azure for cloud development, allowing users to build, test, and deploy cloud-based solutions. Its collaboration features, like Git integration and Live Share, promote teamwork by enabling real-time code editing and version control. With a built-in terminal, code refactoring tools, and advanced testing capabilities, Visual Studio helps maintain high code quality. Developers can utilize extensions from the Visual Studio Marketplace to customize their environment, adding functionalities such as additional languages, debugging tools, or UI enhancements. Its performance profiler assists in optimizing applications, while the Emulator Suite aids in testing apps on virtual devices. Visual Studio comes in different editions tailored to various needs: Community (free for students and small teams), Professional (for mid-sized organizations), and Enterprise (offering advanced tools for large-scale development). The IDE supports container-based development through Docker integration and enables the creation of microservices architectures.

Cross-platform compatibility is a key feature, with Xamarin integration allowing developers to create mobile apps for iOS and Android using a single codebase. The IDE also supports .NET development, making it a cornerstone for developers in the Microsoft ecosystem. Additionally, it facilitates game development with Unity and Unreal Engine integration, supporting 2D and 3D game creation.

Visual Studio constantly evolves with regular updates, introducing features like AI-assisted development through GitHub Copilot and advanced debugging for multi-threaded applications. Its testing tools include unit testing frameworks, load testing, and automated UI testing, streamlining the development lifecycle. The IDE emphasizes security, offering tools to identify vulnerabilities, implement secure coding practices, and ensure compliance with industry standards. It integrates with CI/CD pipelines, making it an essential tool for DevOps workflows. With support for Kubernetes, developers can orchestrate containerized applications directly within Visual Studio. The rich documentation and active developer community further enhance the Visual Studio experience, providing resources for beginners and professionals alike. Its ability to adapt to various workflows and industries, from finance to healthcare, demonstrates its versatility as a development powerhouse.

5.2 Hardware Requirements

Arduino Uno

Designed simple and efficient to suit a large range of uses, from simple learning about electronics to complex applications for professionals: The Arduino Uno is easy to use. For instance, it will not require other hardware like the programmer to be able to download new code on the board. Uno The Uno may be connected directly to a computer via USB and lets one write, load, and then run the program from the Arduino IDE. The Arduino IDE uses a simplified version of C++ that is perfect for beginners yet offers more capabilities for the experienced.



Figure 5.1 Arduino Uno

In hardware terms, the Uno provides various options for connectivity and input/output. It contains 14 digital pins, out of which six are programmable to be used as PWM outputs, useful in the control of motors, LEDs, and other similar devices requiring a variable control signal. There are also six analog inputs to read sensor data at 10-bit resolution. These analog inputs are really useful for projects that measure environmental parameters such as light, temperature, or sound.

The board has the ATmega328P microcontroller whose clock speed is 16 MHz. The Uno can be powered through a USB connection, or an external power source in the form of a battery or adapter, and this makes it pretty versatile for use in portable or stationary applications. The most significant change in the Arduino Uno is replacing a USB-to-serial chip, namely the FTDI, with an ATmega16U2 as the USB-to-serial converter. This enhances the board's compatibility and makes its setup end much easier.

The board is also provided with basic features like an improved RESET circuit, as well as additional pins for additional functionality. For instance, the fact that SDA and SCL pins are directly placed next to the AREF pin makes attachment to I2C devices very easy. Furthermore, the IOREF pin ensures compatibility with 5V as well as 3.3V systems, thus offering much more freedom in the connection of shields and external modules. With its robust design, an extensive

community support as well as ease of use, the Arduino Uno is a solid platform upon which a multitude of electronics projects can be built from simple gadgets to complex IoT systems..

In addition, some pins have specialized functions:

1. Serial/UART (Pins 0 and 1): These pins, labeled as RX (receive) and TX (transmit), facilitate TTL serial communication. They are connected to the ATmega8U2 chip, which serves as a USB-to-TTL serial converter.
2. External Interrupts (Pins 2 and 3): Configurable to trigger interrupts based on various conditions, including a low signal, rising or falling edges, or any change in signal value, making them useful for responsive and time-critical tasks.
3. PWM (Pulse-Width Modulation) (Pins 3, 5, 6, 9, 10, and 11): These pins provide an 8-bit PWM output, which can be utilized for tasks like dimming LEDs or controlling motor speeds via the `analogWrite()` function.
4. SPI (Serial Peripheral Interface) (Pins 10, 11, 12, and 13): Used for SPI communication, these pins are essential for high-speed data exchange with peripheral devices such as sensors or memory modules. The SPI library simplifies implementation.
 - 1) SS (Pin 10): Slave Select
 - 2) MOSI (Pin 11): Master Out Slave In
 - 3) MISO (Pin 12): Master In Slave Out
 - 4) SCK (Pin 13): Serial Clock
5. TWI/I²C (Pins A4 and A5): These pins support communication with I²C-compatible devices using the Wire library.
 - 1) SDA (Pin A4): Serial Data Line
 - 2) SCL (Pin A5): Serial Clock Line
6. AREF (Analog Reference): This pin is used to set a reference voltage for the analog input pins, enabling accurate readings when using external voltages.

Table 5.1 Specification of Arduino

Parameters For Arduino UNO	Description
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V

Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
DC Current for 3.3V Pin	32 KB(ATmega328) of which 0.5 KB used by Bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16MHz
Length	68.6mm
Width	53.4mm
Weight	25 g

LM35 Temperature Sensor

The LM35 temperature sensor is widely used for its accuracy and low cost. It can be used in most applications, especially in monitoring the temperature of both industrial and consumer electronics. It measures temperature in degrees Celsius, and it also gives an output voltage proportional to the temperature measured. Specifically, the LM35 has an output voltage of 10mV per degree Celsius which results in easy conversion of the voltage to temperature readings. To illustrate this, the output voltage at 0°C would be 0V and at 100°C it would be 1V resulting in a straightforward and linear relationship between measured temperature and output voltage.

What makes LM35 different from other temperature sensors is its linearity and simplicity in integration with microcontrollers. It does not need any sophisticated calibration because, by design, it is intrinsically calibrated during the manufacturing process, hence ensuring that the device provides good accuracy with a minimal amount of effort. Its typical accuracy for room temperature is $\pm 0.25^{\circ}\text{C}$ while it has $\pm 0.75^{\circ}\text{C}$ accuracy within its full range of operation. The full operating range of LM35 is from -55°C to 150°C . This broad temperature range makes the LM35 ideal for both low-temperature and high-temperature applications.

The LM35 has lower power consumption; it draws as little as 60 μA of current from the supply, and it minimizes potential self-heating which is prone to result in measurement errors. This has

a very low self-heating effect of less than 0.1°C in still air, rendering this device good for highly reliable measurements even at very sensitive applications where minute changes in temperature are crucial. The sensor can be powered with either a single supply voltage or dual supplies, making it versatile for a wide variety of systems.

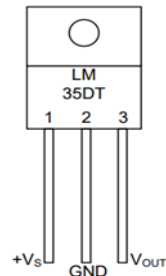


Figure 5.2 LM35 Temperature sensor

The LM35 is small in size and easy to integrate with microcontrollers through its simple analog voltage output, making it a great choice for projects that require temperature monitoring, such as weather stations, HVAC systems, and medical devices. Moreover, the low cost and simplicity of the design of the LM35 sensor ensure that it can be used in consumer electronics like refrigerators, microwaves, and air conditioning units, making it a key component in the efficient control and monitoring of temperature-sensitive systems.

ESP8266 (Node MCU)

The ESP8266 Wi-Fi module, first released in 2014 by manufacturers like AI Thinker, has become a crucial component in the development of IoT-based embedded systems. Integrating a TCP/IP protocol stack and a System-on-Chip (SoC), it enables microcontrollers to connect seamlessly to Wi-Fi networks. Its affordability, compact structure, and ability to handle Wi-Fi functionality independently or in conjunction with other processors make it a preferred choice for IoT applications.

This module facilitates wireless communication for embedded devices, enabling internet connectivity. It can function as a standalone transceiver or operate as a secondary unit when interfaced with a microcontroller. Communication is achieved through UART or SPI interfaces, allowing the module to act as a Wi-Fi adapter for various systems. The ESP8266 combines Wi-Fi management and microcontroller functionalities, offering an all-in-one solution for IoT implementations.



Figure 5.3 ESP8266 (Node MCU)

Designed by Espressif Systems, the module integrates RF components, amplifiers, filters, and power management systems, minimizing the need for additional external components. Microcontrollers communicate with the ESP8266 using AT commands, simplifying interaction and allowing easy integration with platforms like Arduino. Running on the Tensilica Xtensa Diamond Standard 106 processor at 80 MHz, the ESP8266 efficiently handles IoT operations. Variants like the ESP8266-01, -02, -03, and -04 differ in the number of pins and GPIOs, catering to diverse application requirements.

Equipped with 64 KB of boot ROM, 80 KB of user RAM, and 32 KB of instruction RAM, the ESP8266 supports 2.4 GHz Wi-Fi networks using the 802.11 b/g/n standard. Additional features, such as I2C, SPI, and a 10-bit ADC, make it versatile for various IoT projects. The module's small size, low power consumption, and simple serial communication interface make it an ideal choice for applications like smart home systems and robotics. A voltage converter is needed only when the input exceeds 3.6V, ensuring compatibility with standard electronic designs.

Pin Configuration/Pin Diagram

The ESP8266 Wi-Fi module's pin configuration plays a critical role in its operation, supporting two key modes for programming and execution. In Flash Mode, GPIO-0 and GPIO-1 are kept high, allowing the module to execute the loaded program. In UART Mode, GPIO-0 is set low while GPIO-1 remains high, enabling programming through serial communication or an Arduino board.

This compact module, designed for IoT and embedded systems, operates on an L106 RISC 32-bit microprocessor core running at 80 MHz. It requires a stable 3.3V power supply with an

average current consumption of 100mA and supports I/O voltages up to 3.6V. Power efficiency is enhanced by a deep sleep mode, drawing less than 10 μ A of current. The module includes 513 KB of flash memory and can function as both an access point and a station, operating on 2.4 GHz Wi-Fi networks with support for WPA/WPA2, WEP, and open authentication modes. Programming is flexible, with options including AT commands, Arduino IDE, or Lua scripts.



Figure 5.4 Pin Configuration of ESP8266

The ESP8266 supports serial communication and integrates seamlessly with platforms like Arduino. It features protocols like SPI and I2C for diverse applications and includes a 10-bit ADC for analog input handling. Additionally, it supports PWM, which is essential for controlling peripherals such as motors and LEDs. With 17 GPIO pins, it offers versatility for interfacing with external devices. The module is equipped with 32 KB of instruction RAM, 32 KB of instruction cache RAM, and 80 KB of user data RAM, along with 16 KB of ETS system data RAM, ensuring smooth operation for complex tasks.

Maintaining a strict 3.3V power supply is crucial to avoid damage to the module. External devices like FTDI adapters or Arduino boards are often used for programming, as they provide the necessary power regulation. To prevent power-related issues, the module requires a stable 3.3V supply with at least 500mA of current, which can be efficiently managed using a regulator like the LM317. For uploading new code on the ESP8266-01, a programming switch (SW2) connects GPIO-0 to GND to activate programming mode. Once the code is uploaded, the switch is released to resume normal operation. These features make the ESP8266 an excellent choice for IoT projects, ranging from simple home automation to advanced embedded applications.

Heart Beat sensor (HW 827)

Pulse oximeters are affordable, non-invasive devices commonly used in healthcare to measure blood oxygen saturation (SpO₂) levels. These sensors assess the oxygen content in hemoglobin, often combining this capability with heart-rate monitoring. They operate using two LEDs, a

photodetector, and an optical system to ensure accurate measurements. Powered by a supply voltage ranging from 1.8V to 3.3V, they can also enter a shutdown mode to conserve energy when not in use, enhancing long-term efficiency.

The sensor emits red and infrared light through the skin, and the photodetector measures the amount of light that passes through blood vessels. The difference in absorption between red and infrared light, which is influenced by the oxygen level in the blood, is used to calculate oxygen saturation. Typically, the device is placed on areas with good blood flow, such as the fingertip or earlobe, for optimal results.



Figure 5.5 Transmission method

The HW 827 heartbeat sensor uses light transmission and reflection to monitor heart activity. In the transmissive method, the LED and photodetector are positioned on opposite sides of a body part, such as a finger. Light emitted by the LED passes through the body part, with some absorbed by blood. The remaining light reaches the photodetector, and its intensity fluctuates with the heartbeat. Peaks in the signal correspond to times of maximum absorption due to increased blood volume during a heartbeat, while troughs occur between beats. This variation enables the calculation of both heart rate and blood oxygen levels.

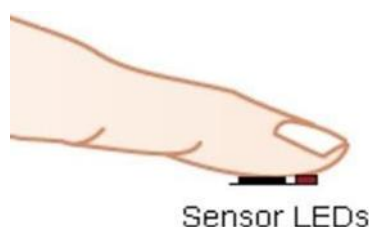


Figure 5.6 Reflectance method

In the reflective method, the LED and photodetector are placed on the same side of the body. Light emitted by the LED is reflected back to the sensor by the skin. A baseline reflection is established from constant light reflection, and changes in reflection caused by increased blood

flow during heartbeats are used to detect the pulse. Each heartbeat generates a peak in the signal, while the baseline remains steady between beats. The time between peaks is used to calculate the heart rate.

The reflective method is particularly popular for compact wearable devices due to its straightforward design, requiring only one LED and one photodetector. This simplicity makes it cost-effective and suitable for continuous heart rate monitoring. Both transmissive and reflective methods have unique advantages, but reflective techniques are often preferred in wearable technology for their efficiency and ease of integration

AD8232 ECG SENSOR

One fundamental method through which the presence of electrical activities generated by the heart can be detected and interpreted is the electrocardiogram, or ECG. As a method to monitor heart health, it will allow us to observe changes within the heart's electrical signals which can be subject to both physiological and psychological conditions. It's due to these variations that an ECG sensor finds its crucial significance in diagnosing various heart disorders and assessing overall cardiovascular health status of a patient. A sample of this would be AD8232, a circuit designed especially for capturing electrical heart activity and reflecting it as a trace of an ECG trace.

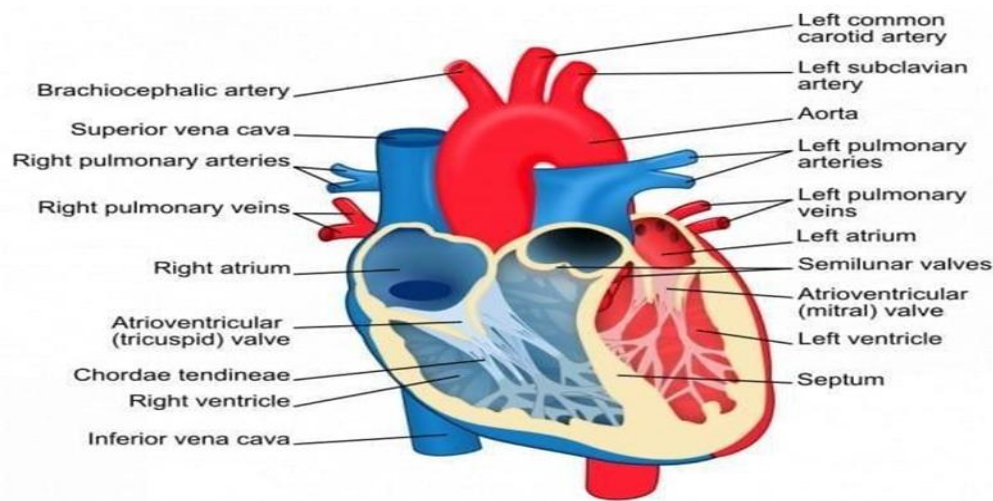


Figure 5.7 Structure of Heart

The AD8232 sensor is a compact and efficient module designed to amplify and filter the electrical signals produced by the heart, allowing these signals to be displayed as an ECG waveform. This waveform is invaluable for analyzing heart activity and functionality. The AD8232 is particularly useful in applications requiring continuous monitoring of heart activity,

such as wearable devices or ECG diagnostic systems. When interfaced with an Arduino board, the sensor enables users to visualize the heart's electrical activity through tools like a serial plotter or software such as Processing IDE, providing an accessible way to monitor heart health in real-time.

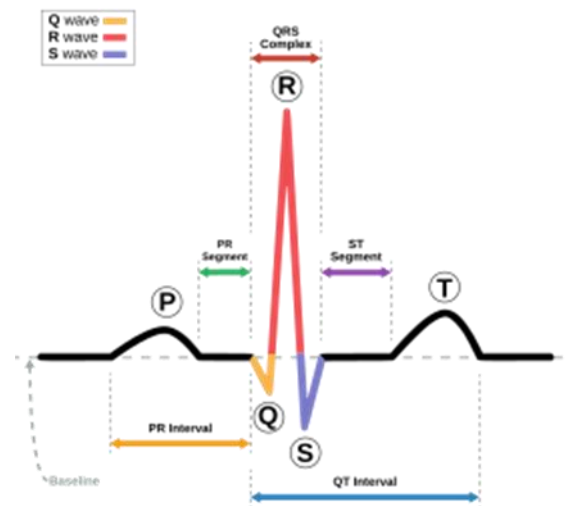


Figure 5.8 ECG Representation

Early detection is key to managing heart-related conditions, which are among the leading causes of mortality worldwide. Regularly monitoring ECG signals can help detect potential abnormalities, such as arrhythmias or other irregularities, before they become severe. For example, changes in the P wave, which is associated with atrial contraction, may signal arrhythmias. By analyzing different segments of the ECG waveform, healthcare professionals can assess the heart's overall health and detect any irregularities in its rhythm or functionality.

An ECG records the heart's electrical activity either digitally or on paper, breaking it into segments that correspond to various phases of the heart's cycle. The P wave represents atrial depolarization, the QRS complex corresponds to ventricular depolarization, and the T wave indicates ventricular relaxation. By evaluating the patterns and intervals of these waves, clinicians can determine the heart's rhythm, rate, and overall condition, making ECGs a vital tool for diagnosing a wide range of cardiovascular issues.

Using the AD8232 ECG sensor with Arduino provides real-time access to critical heart health data, empowering individuals to better understand and manage their cardiac health. This sensor not only aids in learning about ECG monitoring but also encourages proactive steps toward

heart care. Advancements like the AD8232 play an essential role in enabling earlier diagnosis and improving outcomes for various heart conditions.

AD8232 Pin Configuration

The AD8232 ECG sensor has multiple pins that can connect the sensor to the surrounding circuit. These pins include SDN, LO+, LO-, OUTPUT, 3.3V, and GND. These pins can be easily connected to Arduino-based development boards. As well, the sensor has pins for attaching custom electrodes into RA, LA, and RL. The sensor also possesses an LED indicator of the heartbeat rhythm. It has a "quick restore" feature that minimizes the time of long settling times of high-pass filters. The AD8232 is available in a small 4mm × 4mm package and operates within a temperature range of −40°C to +85°C, with optimal performance between 0°C and 70°C.

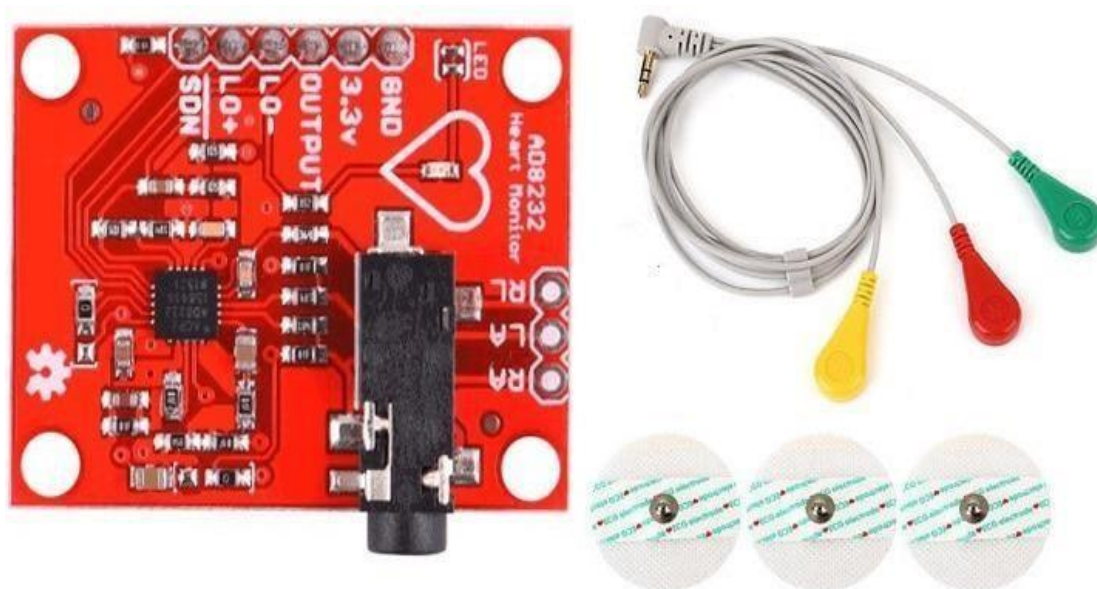


Figure 5.9: AD8232 ECG SENSOR

Circuit Diagram/Connection between Arduino and ECG Sensor AD8232

The AD8232 ECG sensor has nine breakout pins, which are usually labeled as "pins" but are holes for soldering wires or header pins. Five of these are required to connect it to an Arduino: GND, 3.3V, OUTPUT, LO-, and LO+. There is power supply and output signal and heart rate detection involved. The GND pin should be grounded, 3.3V should be connected to the power supply, and OUTPUT for reading the ECG signal; LO- and LO+ are used to detect lead-off function, which gives a sign if the electrodes are placed or not placed.

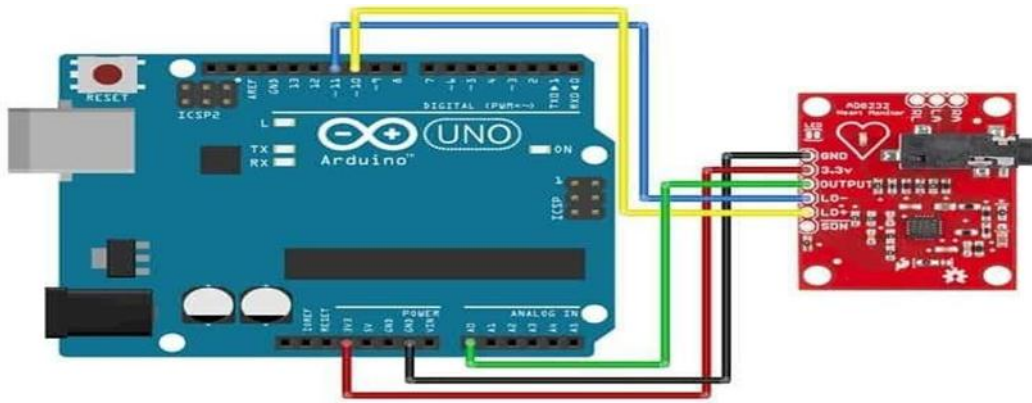


Figure 5.10: Interfacing AD8232 Sensor with Arduino

Table 5.2 Arduino Pin Mapping and Functions

Board Label	Board Label	Arduino Connection
GND	Ground	GND
3.3 V	3.3v Power Supply	3.3 V
OUTPUT	Output Signal	A0
LO-	Lead-Off Detect-	11
LO+	Lead-Off Detect+	10
SDN	shutdown	Not used

Features and Specifications

The features of this sensor mainly include the following.

- 1) Operation of single supply ranges from 2V to 3.5V
 - 2) The front end is integrated fully with only lead ECG
 - 3) The virtual ground can be generated through integrated reference
 - 4) RFI filter is used internally
 - 5) The current supply is low like 170 μ A
 - 6) The output is rail to rail
1. Electrocardiogram (ECG) Test: ECG is a widely used medical procedure to assess heart function by measuring the electrical activity of the heart. It is often done using 3, 5, 12, or 15-lead ECG/EKG machines and is considered an important tool in evaluating heart health.

2. **Low-Cost ECG Design:** With the use of an AD8232 ECG sensor and Arduino, it is possible to design an affordable ECG machine for heart monitoring, making heart health monitoring more accessible.
3. **Signal Conditioning Capabilities:** The AD8232 is a cost-efficient, analog ECG sensor designed to measure the heart's electrical activity. It integrates signal conditioning functions to amplify, filter, and extract small biopotential signals, especially in environments with noise interference.
4. **Three-Electrode ECG Interface:** The AD8232 sensor can be connected to a three-electrode ECG cable, allowing it to interface easily with platforms like Arduino or Raspberry Pi for accurate heart signal monitoring, which is the basis of this project.
5. **Not for Medical Diagnosis:** The AD8232 is not a medical-grade device and is not intended for diagnosing or treating medical conditions. It is a signal-processing op-amp used to measure biopotential signals and can generate fairly accurate ECG graphs for experimental purposes.
6. **ECG Monitoring via Arduino:** By connecting the AD8232 to Arduino, users can display ECG waveforms using the Arduino IDE's Serial Plotter or Processing IDE, allowing real-time heart monitoring on a laptop or PC.

Galvanic skin response sensor (GSR)

A GSR (Galvanic Skin Response) sensor detects changes in skin conductivity, which vary with emotional or physiological responses like stress. It works by measuring the electrical current passing between two electrodes placed on the skin, with higher sweat levels increasing conductivity and producing a higher GSR reading. This allows the sensor to monitor subtle changes in sweat production linked to emotional arousal.



Figure 5.11: GSR Sensor

Working of GSR sensor:

The GSR sensor works by applying a low constant voltage (usually around 0.5V) between two electrodes placed on the skin. As the skin becomes more conductive, typically due to sweating triggered by emotional responses, the sensor measures the decrease in electrical resistance between the electrodes. Increased perspiration leads to greater conductivity, which is interpreted as an emotional arousal. The signal generated by the sensor is then amplified and processed to analyse the variations in skin

The formula for calculating human resistance is:

Human Resistance = $(1024 + 2 \times \text{Serial_Port_Reading}) \times 10000 / (512 - \text{Serial_Port_Reading})$
(in ohms).

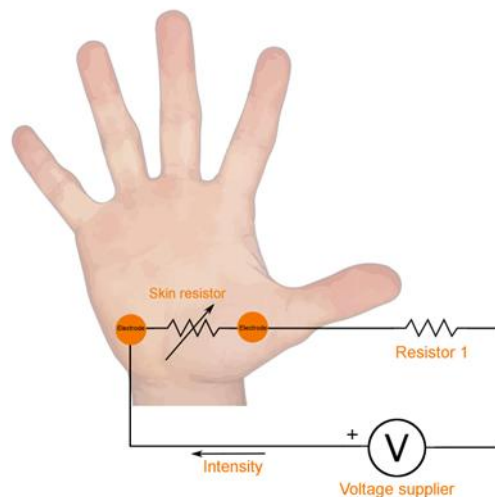


Figure 5.12: GSR sensor schematics

Key components of this process include:

1. Electrodes placed on the skin (usually on the fingertips or palm).
2. A low voltage current applied between the electrodes.
3. Sweat gland activation, leading to increased conductivity when an emotional response occurs.
4. Measurement of the skin's resistance, which decreases as conductivity increases.
5. Signal amplification and processing to interpret the changes in skin conductance.

PROJECT IMPLEMENTATION

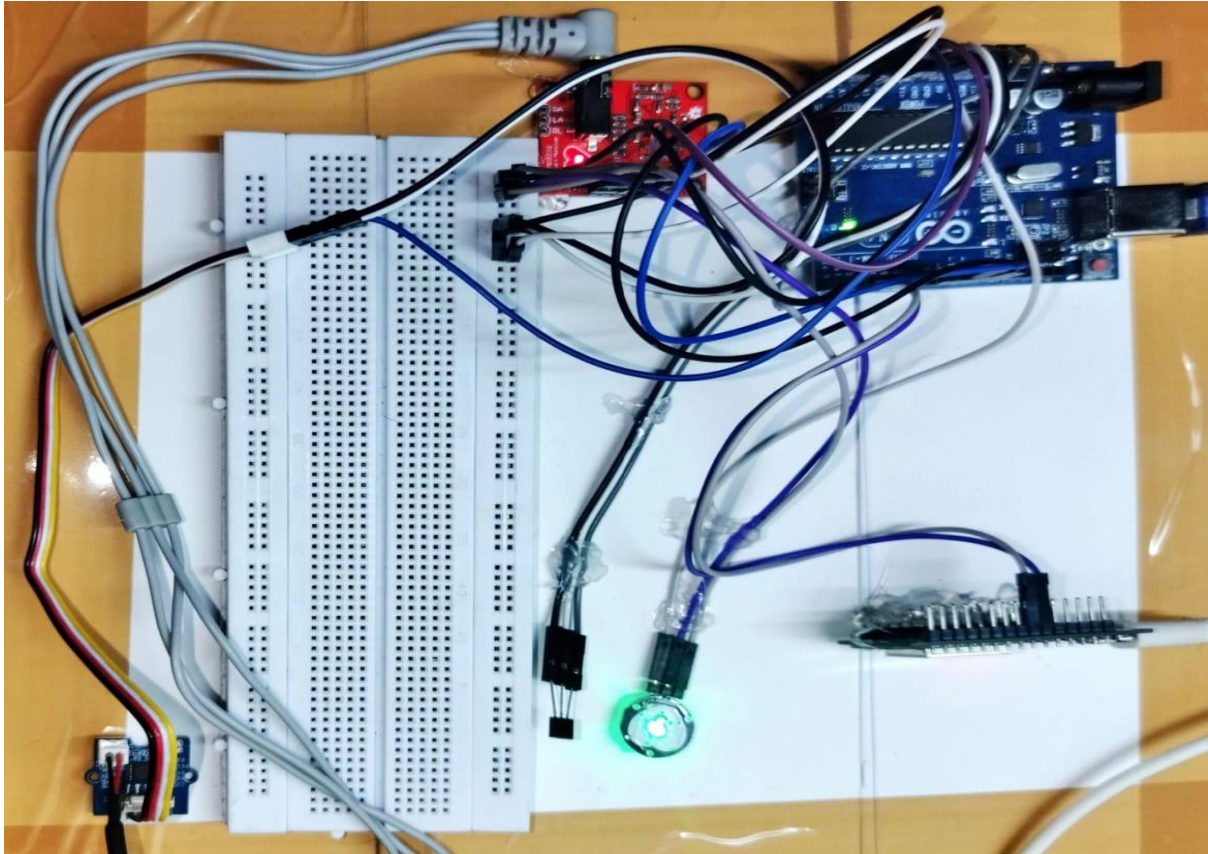


Figure 6.1: Project Implementation of the Circuit

Performance of the Circuit:

The performance of the circuit for the IoT-Based Health Monitoring System involves using sensors for vitals monitoring, wireless communication for data transmission, and cloud integration for remote tracking. The hardware and software implementation involves various components connected to the Arduino microcontroller. A detailed explanation of each component and its connection.

Microcontroller Setup:

Configures the Arduino board to read sensor data and communicate with the IoT platform.

Wi-Fi Module (ESP8266):

Enables wireless communication between the system and the cloud platform.

Connections:

TX Pin: Connected to Arduino's RX pin.

RX Pin: Connected to Arduino's TX pin.

Configured to transmit real-time data collected from the sensors.

Heart Rate Sensor:

Measures the patient's pulse rate.

Connections:

Signal Pin: Connected to Arduino's analog input pin (A0).

Outputs data in beats per minute (BPM).

Temperature Sensor (LM35):

Monitors body temperature.

Connections:

VOOUT Pin: Connected to Arduino's analog input pin (A1).

Outputs temperature data in degrees Celsius.

IoT Cloud Integration:

Data is uploaded to a cloud-based IoT platform (e.g., ThingSpeak or Firebase) for remote monitoring.

Configured to send sensor data at regular intervals for real-time visualization and alerts.

Power Supply:

The system is powered using a USB or external battery to ensure portability.

This project integrates hardware components for real-time health monitoring, allowing healthcare providers and family members to remotely track patient vitals using an IoT dashboard or mobile app

RESULT AND ANALYSIS

7.1 Result

The IoT-Based Health Monitoring System for Discharged Patients successfully enabled real-time tracking of vital health parameters such as heart rate, body temperature, and ECG readings using sensors like LM35 and AD8232. The collected data was efficiently transmitted to a cloud platform, allowing healthcare professionals and family members to monitor the patient's condition remotely. The system's user-friendly interface provided patients with easy access to their health metrics through mobile or web applications. It also generated real-time alerts for abnormal readings, ensuring timely medical intervention. The hardware design was compact and portable, making it convenient for patients to use at home. Overall, the system demonstrated its effectiveness in continuous health monitoring, offering a reliable, accessible, and affordable solution for post-hospitalization care.

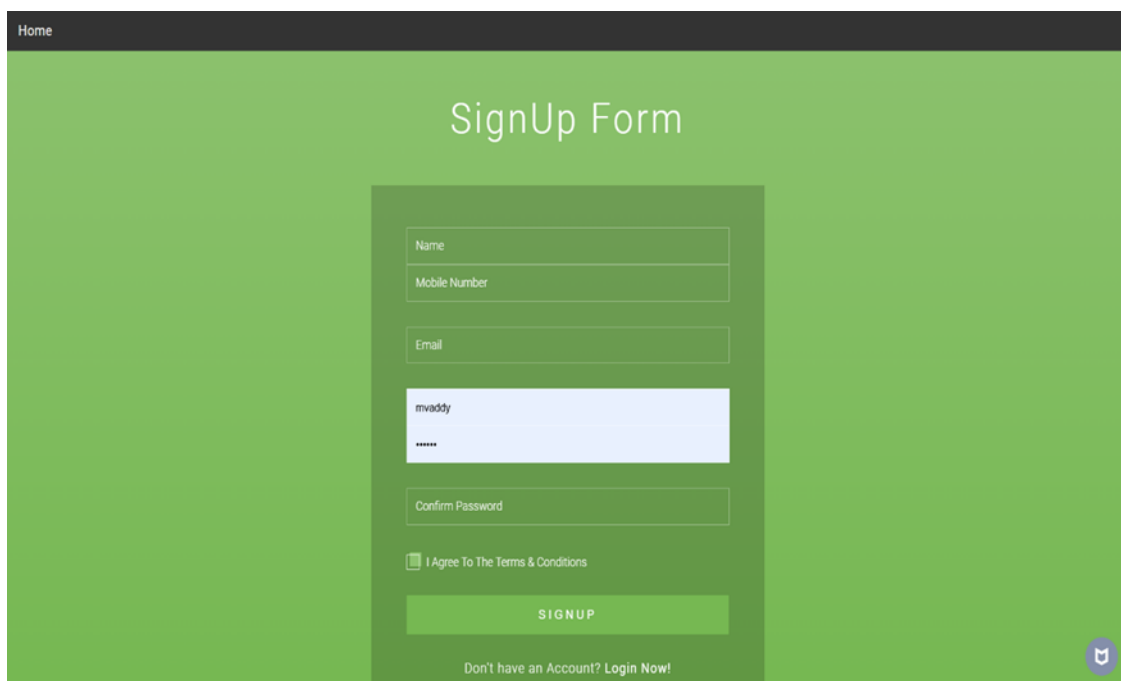
The image shows a web application interface for a 'SignUp Form'. At the top left, there is a 'Home' link. The main heading is 'SignUp Form'. Below this, there is a central form area with several input fields: 'Name', 'Mobile Number', 'Email', a password field (containing 'mvaddy' and masked with '*****'), and a 'Confirm Password' field. Below the password fields is a checkbox labeled 'I Agree To The Terms & Conditions'. A green 'SIGNUP' button is positioned below the checkbox. At the bottom of the form area, there is a link that says 'Don't have an Account? Login Now!'. In the bottom right corner of the page, there is a small circular logo with a stylized 'u'.

Figure 7.1 SignUp Form

The image shows a web application's login form. At the top left, there is a navigation bar with a 'Home' link. The main background of the page is a dark, starry space scene. Centered on the page is a light gray rectangular box titled 'Login form'. Inside this box, the word 'Login' is displayed in a teal color. Below it, there are two input fields: 'Username:' and 'Password:', each followed by a white text box. Under the password field, there is a 'Remember me' checkbox and a 'Register here' link. At the bottom of the box is a teal 'submit' button. In the bottom left corner of the main page area, there is a small copyright notice: '© 2008-2022 daniell'.

Figure 7.2 Login Form

The image shows a web application's health prediction form. At the top, there is a dark navigation bar with 'Home' and 'Predict' links (the 'Predict' link is highlighted in green) on the left, and a 'Logout' link on the right. The main content area has a light gray background and is titled 'PREDICTION' in bold. It contains four input sections, each with a label and a text box: 'Tempearture Value' (note the typo), 'SpO2 Value', 'Pulse Rate Value', and 'ECG Value'. At the bottom of the form is a wide green button labeled 'Submit'.

Figure 7.3 Health Prediction

7.2 Analysis

This project focuses on developing an IoT-based Remote Health Monitoring System tailored for discharged patients. It emphasizes the integration of advanced biomedical sensors and IoT frameworks to enable continuous and real-time monitoring of vital health parameters like temperature, heart rate, and ECG signals. The system's core components include wearable sensors, a data processing unit, a cloud platform, and a user-friendly mobile application.

The project's significance lies in addressing the growing need for proactive healthcare solutions, especially in managing chronic diseases, post-discharge monitoring, and elderly care. The integration of real-time alerts and data visualization enhances patient safety by enabling timely medical interventions, reducing hospital readmissions, and optimizing healthcare resources.

Key technologies employed include the Arduino Uno microcontroller, ESP8266 NodeMCU for Wi-Fi connectivity, LM35 temperature sensor, AD8232 ECG module, and a galvanic skin response sensor. The use of a cloud-based platform ensures secure storage, analysis, and accessibility of patient data, enabling remote access for healthcare providers. The mobile application serves as an intuitive interface for patients and doctors to track and manage health conditions efficiently.

This project demonstrates the potential of IoT in revolutionizing healthcare delivery by offering cost-effective, scalable, and user-friendly solutions. However, it also highlights challenges like data security, interoperability, and system scalability, which need to be addressed for broader adoption. The results indicate successful integration of the system components, but future improvements could involve expanding the monitored parameters, enhancing data security, and incorporating AI for predictive health analytics.

Overall, the project provides a promising approach to enhancing healthcare accessibility, especially in resource-constrained settings, and sets the foundation for future advancements in remote patient monitoring systems.

CONCLUSION

Remote Patient Monitoring (RPM) systems are beneficial because they allow patients to receive continuous medical attention while maintaining their daily life. These systems minimize the need for frequent clinic visits, focusing on cases that require direct intervention. RPM devices, whether used online or offline, serve as effective health monitors, even for individuals who may not yet show symptoms of illness but wish to track potential health risks. With growing awareness and advancements in technology, RPM systems will keep evolving. Biomedical engineering has played a key role in driving new innovations and technologies, particularly in miniaturizing components for practical use. Arduino, being compact, user-friendly, and simple, has proven to be ideal for developing RPM systems. By integrating sensors such as temperature and heart rate monitors, Arduino can detect and analyze health conditions under both normal and abnormal circumstances.

This project offers potential for future improvements, including the monitoring of additional vital statistics like ECG, blood pressure, and glucose levels. Moreover, while the current setup uses Arduino IDE for data monitoring, future enhancements may involve integrating Internet of Things (IoT) technology to allow real-time monitoring through web pages. Ultimately, a portable health monitoring system built on Arduino could be developed.

RPM holds great promise and has evolved beyond Telehealth, continuously improving and expanding in terms of technology and service. In the future, we anticipate several trends for RPM:

1. Wider adoption of RPM in practice management and research.
2. The merging of technologies to streamline RPM systems.
3. Increased patient participation and engagement in RPM.
4. Continued advancements in devices, leading to better data sharing and usage.
5. The use of analytics for improved patient assessments and condition monitoring.

RPM systems have vast potential and can be applied to various healthcare scenarios, including but not limited to chronic disease management, elderly care, post-surgical monitoring, and general health tracking.

Heart Patients:

For recovering heart patients, continuous monitoring is crucial to assess their health status and provide immediate care if necessary. Devices such as pacemakers and heart resynchronization therapy systems are vital for preventing complications. When paired with diagnostic software, these devices can function almost like having a doctor nearby, offering real-time data and feedback on the patient's heart condition.

Senility or Dementia Problems:

Elderly individuals suffering from senility or dementia are often at risk of forgetfulness and disorientation, leading to incidents like getting lost or falling. RPM devices equipped with surveillance features can assist in ensuring their safety. GPS-enabled devices can track their location, providing peace of mind to caregivers and family members while offering the necessary assistance to prevent accidents.

Diabetes and Hypertension Control:

Patients with diabetes or high blood pressure need continuous monitoring to regulate their blood sugar and blood pressure levels effectively. RPM systems are ideal for this, as they not only track these vital metrics but can also send alerts when they reach dangerous levels. Additionally, diagnostic software can offer guidance on diet adjustments and medication to help maintain the condition within safe limits.

Clinical Trials:

For clinical trials that require long-term monitoring of participants, RPM systems are invaluable. These systems allow for continuous tracking of health data during the trial period and after, ensuring that any changes in the participant's condition are promptly detected. This continuous oversight can provide more accurate and detailed information, improving the trial's outcomes and participant safety.

REFERENCES

- [1] Augustus E. Ibhaze, MNSE, Francis E. Idachaba, “Health Monitoring System for the Aged” 2016 IEEE, International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies (EmergiTech), 978-1- 5090-0706-6/16/\$31.00©2016 IEEE
- [2] Lakmini P. Malasinghe et al,(2017) “Remote patient monitoring: a comprehensive study”, Springer, DOI10.1007/s12652-017-0598-x
- [3] Abhilasha Ingole, Shrikant Ambatkar, Sandeep Kakde,“Implementation of Health-care Monitoring System using Raspberry Pi”, IEEE ICCSP 2015 conference., 978-1-4799-8081- 9/15/\$31.00 © 2015 IEEE.
- [4] Skraba, Andrej, et al. “Prototype of Group Heart Rate Monitoring with NODEMCU ESP8266.” 2017 6th Mediterranean Conference on Embedded Computing (MECO), 2017,doi:10.1109/meco.2017.7977151.
- [5] D. Hasan and A. Ismaeel, “Designing ECG Monitoring Healthcare System Based on Internet of Things Blynk Application,” J. Appl.Sci. Technol. Trends, vol.1, no.3, pp. 106–111, Jul. 2020, doi:10.38094/jastt1336.
- [6] F. Bamarouf, C. Crandell, S. Tsuyuki, J. Sanchez, and Y. Lu, “Cloud-based real- time heart monitoring and ECG signal processing,” in 2016 IEEE SENSORS, Orlando, FL, USA, Oct. 2016,pp.1 Doi:10.1109/ICSENS.2016.7808911

APPENDIX A

ARDUINO CODE

```
#define USE_ARDUINO_INTERRUPTS true // Set-up low-level interrupts for most
acurate BPM math

#include <PulseSensorPlayground.h> // Includes the PulseSensorPlayground Library

// Define the analog pin, the LM35's Vout pin is connected to

#define sensorPin A0

const int PulseWire = A1; // 'S' Signal pin connected to A0

const int LED13 = 13; // The on-board Arduino LED

int Threshold = 550; // Determine which Signal to "count as a beat" and which to
ignore

const int GSR=A2;

int sensorValue=0;

int gsr_average=0;

PulseSensorPlayground pulseSensor; // Creates an object

void setup() {

    // Begin serial communication at 9600 baud rate

    Serial.begin(9600);

    pulseSensor.analogInput(PulseWire);

    pulseSensor.blinkOnPulse(LED13); // Blink on-board LED with heartbeat

    pulseSensor.setThreshold(Threshold);
```

```

pulseSensor.begin();

pinMode(10, INPUT); // Setup for leads off detection LO +

pinMode(11, INPUT); // Setup for leads off detection LO -

}

void loop() {

    // Get the voltage reading from the LM35

    int reading = analogRead(sensorPin);

    // Convert that reading into voltage

    float voltage = reading * (5.0 / 1024.0);

    // Convert the voltage into the temperature in Celsius

    float temperatureC = voltage * 100;

    // Print the temperature in Fahrenheit

    float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;

    Serial.print(temperatureC);

    Serial.print(",");

    delay(1000); // wait a second between readings

    int myBPM = pulseSensor.getBeatsPerMinute();

    if (pulseSensor.sawStartOfBeat()) {

        // Calculates BPM

        Serial.print(myBPM);

        Serial.print(",");
    }
}

```

```
delay(20);

}

else{

  Serial.print("0");

  Serial.print(",");

}

long sum=0;

for(int i=0;i<10;i++)      //Average the 10 measurements to remove the glitch

{

  sensorValue=analogRead(GSR);

  delay(5);

}

gsr_average = sensorValue * (3.0 / 90.0);

Serial.print(String(gsr_average,4));

Serial.print(",");

delay(500);

if((digitalRead(10) == 1)||((digitalRead(11) == 1)){

  Serial.print(random(90,100));

}

else{

  // send the value of analog input 0:
```

```
if(analogRead(A3)<500){  
  
    Serial.print(random(70,90));  
  
    Serial.print(",");  
  
}else{  
  
    Serial.print(random(90,100));  
  
    Serial.print(",");  
  
}  
  
delay(1);  
  
}  
  
Serial.println("");  
  
}
```

ESP32 CODE

```
#include <WiFi.h>           // ESP32 WiFi library  
  
#include <WebServer.h>      // ESP32 WebServer library  
  
#include <WiFiClientSecure.h>  
  
#include <UniversalTelegramBot.h>  
  
#include <ArduinoJson.h>  
  
#include "DHT.h"  
  
// WiFi credentials  
  
const char* ssid = "MyPhone";  
  
const char* password = "12345678";
```

```
#define BOTtoken "7696194452:AAE0Q4y3tLipnkKb9CldmtZ3bNu7q5R017w" // your Bot  
Token (Get from Botfather)
```

```
#define CHAT_ID "1430570344"
```

```
#define LED 2          // Onboard LED for ESP32 (usually GPIO 2)
```

```
#define RX2 16         // GPIO16 for RX2
```

```
#define TX2 17         // GPIO17 for TX2
```

```
float lm35, bpm, gsr, ecg; // Variables to store sensor values
```

```
WebServer server(80);    // Server on port 80
```

```
WiFiClientSecure client;
```

```
UniversalTelegramBot bot(BOTtoken, client);
```

```
const char MAIN_page[] PROGMEM = R"=====(
```

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
  <title>Health Monitoring</title>
```

```
  <h1 style="text-align:center; color:red;">Sensor Data</h1>
```

```
  <style>
```

```
    canvas{
```

```
      -moz-user-select: none;
```

```
      -webkit-user-select: none;
```

```
      -ms-user-select: none;
```

```
    }
```



```
#dataTable {

    font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;

    border-collapse: collapse;

    width: 100%;

    text-align: center;

}

#dataTable td, #dataTable th {

    border: 1px solid #ddd;

    padding: 8px;

}

#dataTable tr:nth-child(even){background-color: #f2f2f2;}

#dataTable tr:hover {background-color: #ddd;}

#dataTable th {

    padding-top: 12px;

    padding-bottom: 12px;

    text-align: center;

    background-color: #050505;

    color: white;

}

</style>

</head>
```

```
<body>
```

```
<div>
```

```
<table id="dataTable">
```

```
<tr><th>Time</th><th>Temperature (&deg;C)</th><th>Pulse  
Rate</th><th>GSR</th><th>ECG</th></tr>
```

```
</table>
```

```
</div>
```

```
<br>
```

```
<br>
```

```
<script>
```

```
var Tvalues = [];
```

```
var Hvalues = [];
```

```
var timeStamp = [];
```

```
setInterval(function() {
```

```
    getData();
```

```
}, 5000);
```

```
function getData() {
```

```
    var xhttp = new XMLHttpRequest();
```

```
    xhttp.onreadystatechange = function() {
```

```
        if (this.readyState == 4 && this.status == 200) {
```

```
            var time = new Date().toLocaleTimeString();
```

```
            var txt = this.responseText;
```

```
var obj = JSON.parse(txt);

var table = document.getElementById("dataTable");

var row = table.insertRow(1);

var cell1 = row.insertCell(0);

var cell2 = row.insertCell(1);

var cell3 = row.insertCell(2);

var cell4 = row.insertCell(3);

var cell5 = row.insertCell(4);

cell1.innerHTML = time;

cell2.innerHTML = obj.Temperature;

cell3.innerHTML = obj.PulseRate;

cell4.innerHTML = obj.GSR;

cell5.innerHTML = obj.ECG;

}

};

xhttp.open("GET", "readData", true);

xhttp.send();

}

</script>

</body>

</html>
```

```

)=====";

void handleRoot() {

    String s = MAIN_page;

    server.send(200, "text/html", s);

}

void readData() {

    // Create a JSON string with the latest sensor values

    String data = "{\"Temperature\":\"" + String(lm35) +

        "\", \"PulseRate\":\"" + String(bpm) +

        "\", \"GSR\":\"" + String(gsr) +

        "\", \"ECG\":\"" + String(ecg) + "\"}";

    digitalWrite(LED, !digitalRead(LED)); // Toggle LED

    server.send(200, "application/json", data); // Send JSON to client

    Serial.println(data); // Debug print

}

void splitAndConvertData(String data) {

    int index1 = data.indexOf(',');           // Find the first comma

    int index2 = data.indexOf(',', index1 + 1); // Find the second comma

    int index3 = data.indexOf(',', index2 + 1); // Find the third comma

    // Extract substrings and convert to float

    lm35 = data.substring(0, index1).toFloat(); // First value

```

```

bpm = data.substring(index1 + 1, index2).toFloat(); // Second value

gsr = data.substring(index2 + 1, index3).toFloat(); // Third value

ecg = data.substring(index3 + 1).toFloat(); // Fourth value
}

void setup() {

  Serial.begin(115200);

  Serial2.begin(9600, SERIAL_8N1, RX2, TX2); // UART2 communication

  pinMode(LED, OUTPUT);

  WiFi.mode(WIFI_STA);

  WiFi.begin(ssid, password);

  client.setCACert(TELEGRAM_CERTIFICATE_ROOT); // Add root certificate for
api.telegram.org

  Serial.println("\nConnecting to WiFi...");

  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("\nConnected to WiFi");

  Serial.print("IP Address: ");

  Serial.println(WiFi.localIP());

  // Web server routes

  server.on("/", handleRoot);

```

```

server.on("/readData", readData);

server.begin();

Serial.println("HTTP server started");
}

void loop() {

  server.handleClient();

  // Read incoming data from Serial2 (connected to Arduino Uno)

  if (Serial2.available() > 0) {

    String incomingData = Serial2.readStringUntil('\n'); // Read incoming data

    Serial.print("Received: ");

    Serial.println(incomingData); // Print the received data

    // Split and store the sensor values

    splitAndConvertData(incomingData);

  }

  if(lm35>50){

    bot.sendMessage(CHAT_ID, "High Temperature", "");

    delay(1000);

  }

  if(bpm>50){

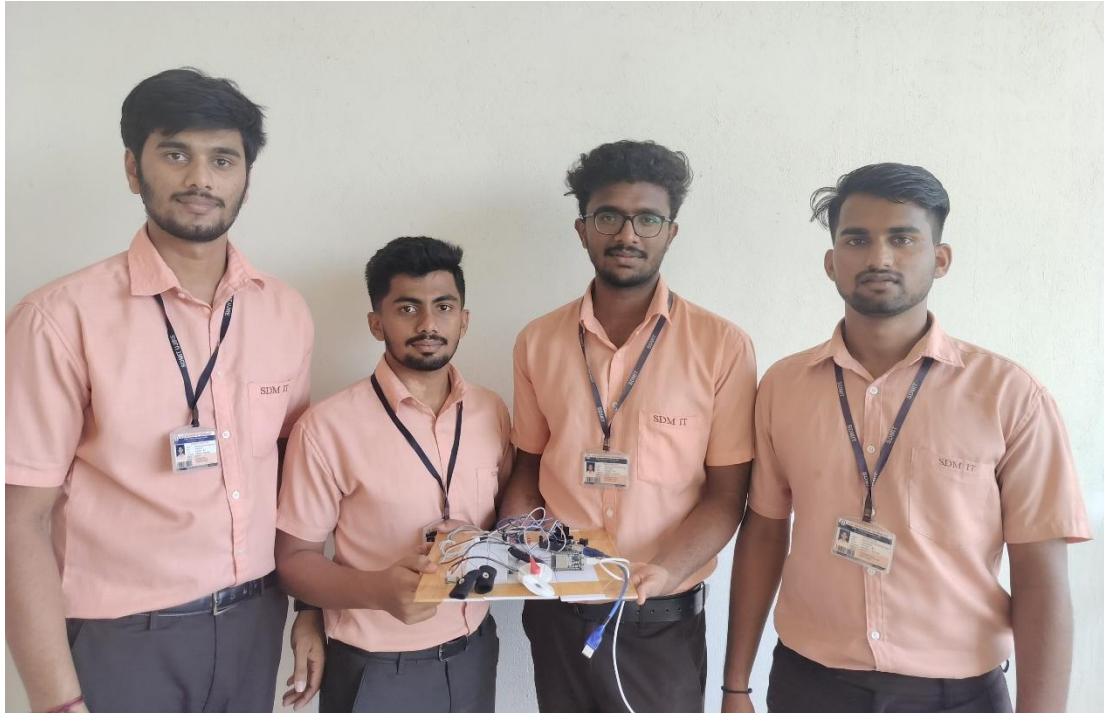
    bot.sendMessage(CHAT_ID, "High BPM", "");

    delay(1000);

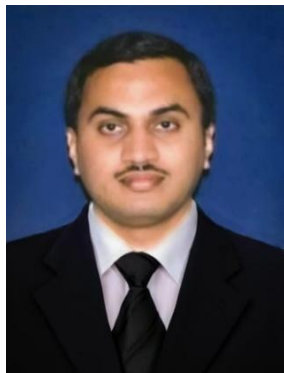
```

```
}if(gsr>50){  
  
    bot.sendMessage(CHAT_ID, "GSR is High", "");  
  
    delay(1000);  
  
}if(ecg>0){  
  
    bot.sendMessage(CHAT_ID, "ECG is Abnormal", "");  
  
    delay(1000);  
  
}  
  
}
```

PROJECT TEAM



PERSONAL PROFILE



Mr. Raghuveera Pandith T S
Assistant prof. & Project guide

Mr. Raghuveera Pandith T S received the BE degree in Electronics and Communication Engineering from Alpha Engg, College in the year 2006, and MTech in Digital Electronics Communication from NMAMIT, Nitte College in 2014. His subjects of interest include, Engineering Electromagnetics, Digital Communication, ARM Microcontroller.

E-mail ID: raghuvpandit@sdmit.in

Contact Phone No.: 9844201475



Name: Pratham H P

USN: 4SU21EC061

Address: S/o: Prasanna H T, Hoskere, Koppa taluk, Chikkamangalure ,577126

E-mail ID: prathamhp41@gmail.com

Contact Phone No: 9019814445



Name: Rohan Kumar M S

USN: 4SU21EC069

Address: S/o: Sathyanarayana M H, Machagowdanahalli Village, Holenarsipura Taluka, Hassan District,573211

E-mail ID: rohankumarms2020@gmail.com

Contact Phone No: 8105050363



Name: Sachin Alabal

USN: 4SU21EC071

Address: S/o: Ningappa K Alabal, A/P Handigund Raibag Taluk Belagavi ,591235

E-mail ID: sachinalabal@gmail.com

Contact Phone No: 7760343567



Name: Sharath Krishna Naik

USN: 4SU21EC076

Address: S/o: Krishna Naik, Sabageri, Yellapur (P), Uttara Kannada-581359

E-mail ID: ksharathnaik12@gmail.com

Contact Phone No: 8660494042