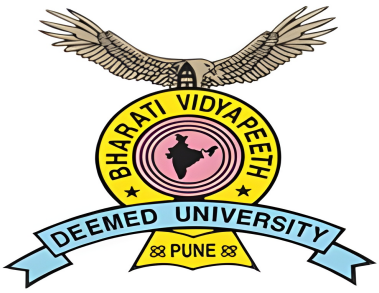


A

Project Stage-II Report on

Body Posture Detection using Wearable Sensors and Machine Learning

Submitted in the partial fulfillment of the requirements of Semester-VIII

For the Award of the Degree of Bachelor of Technology (B.Tech) in

Electronics and Communication Engineering

Submitted by

Bhavya Jain	**PRN:**	**2114110458**
Satyam Mishra	**PRN:**	**2214110605**
Ram Prakash	**PRN:**	**2214110681**

Under the Guidance of

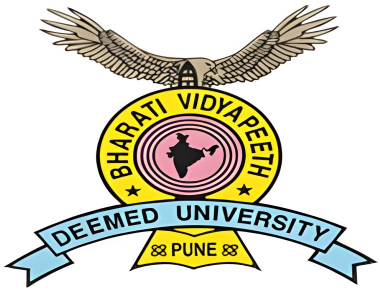
Prof Dr. Tanuja S. Dhope

Department of Electronics and Communication Engineering

BHARATI VIDYAPEETH DEEMED TO BE UNIVERSITY

COLLEGE OF ENGINEERING, PUNE - 411043

SEMESTER-VIII, ACADEMIC YEAR: 2024-2025



Department of Electronics and Communication Engineering
BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY)
COLLEGE OF ENGINEERING, PUNE

CERTIFICATE

This is to certify that the project phase-II report on **"Body Posture Detection using Wearable Sensors and Machine Learning "** submitted by

Bhavya Jain	**2114110458**
Satyam Mishra	**2214110605**
Ram Prakash	**2214110681**

in partial fulfillment of the requirements for the award of degree of (B.Tech) Bachelor of Technology in Electronics and Communication Engineering.

Prof. Dr. Tanuja S. Dhope
Guide, ECE Dept.,
BV(DUCOE), Pune

Dr. Dhiraj M. Dhane
Project Co-ordinator
BV(DUCOE), Pune

Prof.(Dr.) Arundhati A. Shinde
Head of the Department
BV(DUCOE), Pune

Date:

Place: Pune

Abstract

Poor posture is a common issue in today's digital and sedentary age, often leading to discomfort, spinal misalignment, and long-term health problems. This project presents a real-time Body Posture Analyzing System that employs eleven MPU9250 sensors placed at critical body positions including the head, neck, shoulders, upper and lower back, knees, and ankles. These sensors gather gyroscopic and accelerometric data to track orientation and movement. Two ESP32 microcontrollers act as slaves to collect sensor data and relay it to a master ESP32 via SPI communication, ensuring efficient and synchronized data handling.

The system aims to accurately detect deviations from ideal posture and provide analytical insight, which can be particularly beneficial in clinical diagnostics, physical rehabilitation, workplace ergonomics, and athletic training. Its modular and scalable hardware configuration ensures flexibility for various user needs. Preliminary tests reveal the system's consistency in posture recognition and its potential for integration with machine learning models for real-time feedback and correction. The approach lays a strong foundation for smart health monitoring solutions that promote long-term physical well-being.

Acknowledgements

We extend our heartfelt gratitude to Dr. Tanuja S. Dhope, our mentor, for her invaluable guidance, support, and encouragement throughout this project. Her expertise and dedication have greatly influenced the success of our work. We also thank Prof. (Dr.) Arundhati A. Shinde, Head of the Department, for her continuous encouragement and leadership, which provided us with the confidence to pursue this project with enthusiasm.

We are deeply grateful to our department and institution for the opportunity and resources provided, enabling us to carry out this research effectively.

Bhavya Jain

Satyam Mishra

Ram Prakash

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Posture correction stands essential for Muscloskeletal Disease (MSD) prevention because modern populations lead sedentary lives while ergonomics rapidly decline yet millions are at risk worldwide [1]. The improper position of our body generates painful results coupled with decreased mobility along with fatigue which cause permanent physical changes that affect quality of life negatively [2]. The economic burden for MSDs results from healthcare expenses combined with lost workdays and diminished performance at work. Standard approaches to posture assessment through physiotherapy and mirror biofeedback lack standardized procedures and demonstrate variable results that limit scalability while not providing real-time monitoring. A rising demand exists for intelligent automatic posture detection systems that address current monitoring deficits. Real-time body movement tracking and posture monitoring expands through IoT networks because of modern sensor technology success particularly with inertial measurement units (IMUs). The system provides affordable scalability along with real-time posture data analysis followed by storage in the cloud infrastructure for simple access. The proposed research develops a multi-sensor body posture analyzer through implementation of MPU9250 sensone sensors which are strategically positioned across the entire body structure. The body movement is recorded by these sensors which incorporate their built-in accelerometers and gyroscopes and magne-

tometers [3]. A NodeMCU ESP32S microcontroller facilitates data acquisition and communication via SPI. Data moves from the sensors to an MQTT-based cloud dashboard that allows instant posture monitoring with visualization capabilities.

This system serves three main user groups including individuals as well as hospitals and fitness trainers while providing ergonomics consultants with beneficial services. The device serves both healthcare facilities and rehabilitation centers and workplace ergonomics and sports environments to become an essential tool that advances lifestyle health and overall well-being.

1.2 Overview

People worldwide now face major health problems stemming from improper body posture because society continues to transition towards sedentary lifestyles coupled with desk-based work environments. Poor posture triggers musculoskeletal disorders (MSDs) that manifest as chronic pain symptoms alongside reduced mobility while creating fatigue and causing permanent body tissue adjustments. NBSDs result in substantial economic tolls through direct healthcare expenses together with diminished work time output at the workplace. Traditional posture monitoring methods, such as mirror biofeedback, physiotherapy analysis, and apparatus-based evaluations, have limitations:

Subjectivity and Variability: Human opinion-based evaluation produces results that are not reliable because they are inconsistent. Lack of Real-time Feedback: Many standard measurement techniques fail to function effectively during periods of dynamic body posture. Scalability Issues: The approaches prove extremely slow and cannot scale for large-scale implementations. Recent advances in wearable systems alongside Internet of Things technology have generated novel chances for instant posture monitoring. Cloud-based data processing combined with wearable sensors enables nonstop body movement tracking followed by instantly actionable feedback. New technological advancements handle the limitations of conventional approaches through adaptable solutions which provide stable and scalable applications in ergonomic settings and

healthcare programs.

1.3 Problem Statement

While innovative posture detection systems exist, they face significant challenges: Traditional systems create subjective results while humans make mistakes. The availability of real-time monitoring along with corrective feedback remains scarce. The current implementation of technologies requires a high financial investment alongside limited scalability which leaves typical users unable to adopt them. The technology landscape demands affordable and scalable as well as reliable real-time body posture analysis systems featuring modern sensors alongside machine learning methods to deliver precise actionable feedback.

1.4 Objectives

The primary objectives of this research are:

- Develop a Cost-effective and Scalable System: The affordable hardware selection of MPU9250 and ESP32 enables implementation at larger volumes and enhances universal adoption.
- Real-time Monitoring and Feedback: Through the implementation of direct IoT communication protocols SPI together with MQTT users can create rapid data-sharing networks that immediately adjust postures.
- Enhance Accuracy Using Machine Learning: The system uses machine learning algorithms to recognize postures through support vector machines parameterized by artificial neural networks.
- Cloud-based Dashboard: Users can assess data using an approachable interface which lets them track data and study details remotely.

Chapter 2

Literature review

2.1 Overview

Several studies and projects have been conducted on the application of IMU sensors and microcontrollers in posture monitoring systems, which have led to the development of effective solutions for tracking body posture:

- IMU-Based Posture Monitoring: Initiators known as MPU9250 unite accelerometers with gyroscopes along with magnetometers to function in wearable tools for detailed body position recording and motion observation. The combination of such sensors presents an established method to accurately monitor angular position change and posture malcorrected movement. Research undertaken by Parisi et al. (2020) proved IMUs function effectively to track the postures for the upper part of the human body and indicate typical alignment flaws so they feature as excellent candidates for ongoing posture assessment.

- Microcontroller-Based Posture Tracking: The posture tracking system relies on ESP32 microcontrollers as these devices integrate low power consumption with Wi-Fi and Bluetooth connectivity and can be easily implemented with various sensors. According to Kwon et al. (2019), the ESP32 shows excellent performance in long-term continuous tracking applications for health by acquiring and transmitting real-time data functions.

- IoT and Data Transmission Protocols: High-speed data transfers occur through SPI device communication but MQTT accomplishes efficient real-time message transfers to cloud-based systems. The study by Patel et al. (2021) demonstrated how MQTT facilitated IoT health monitoring by providing enhanced data access with remote monitoring and better user engagement required to obtain effective posture correction outcomes.

- Real-time non-invasive posture monitoring technology development has focused on methods to prevent and address musculoskeletal disorders (MSDs) because of their requirements for immediate feedback and broad adoption possibilities. Traditional posture correction methods including physiotherapy evaluation and biofeedback show limitations by being subjective together with their restricted scalability and delay in obtaining feedback. Sensor-based systems integrated with IoT frameworks make it possible to automatically monitor assets continuously over remote environments while also delivering actionable understanding. Today's posture monitoring functions better thanks to machine learning (ML) techniques which apply automated data analysis and posture anomaly detection to provide data-centric personalized posture correction models. Through their union with wearable technology and IoT and machine learning systems researchers have developed performance-driven solutions which offer cost-efficient scalability for posture analysis.

2.2 Related Work

1. Dobrea Dobrea (2018) [4]: "A Warning Wearable System Used to Identify Poor Body Postures"

This study introduces Intell.TieSens, a wearable device designed to detect and alert users of poor body posture in real-time. The system is portable, lightweight, and features low power consumption with Bluetooth wireless communication capabilities. By monitoring body alignment, it provides immediate feedback to prevent lumbar and

cervical pain associated with prolonged poor posture. The authors detail the system's architecture, sensor integration, and potential applications in occupational health and daily life. Preliminary evaluations suggest that Intell.TieSens effectively identifies posture issues, offering a proactive solution for posture-related health problems.

2. Nistane et al. (2021) [5]: "An IoT-Based Rectification of Posture Using Biofeedback"

This research presents an Internet of Things (IoT)-based system aimed at correcting human posture through biofeedback mechanisms. The device monitors user posture and provides real-time feedback via vibrations, facilitating immediate correction. The system's architecture includes sensors that detect posture deviations and a feedback mechanism to alert users. The authors discuss the implementation of the system and its potential to address back problems resulting from incorrect posture, emphasizing its relevance in promoting spinal health.

3. Severin (2020) [6]: "The Head Posture System Based on 3 Inertial Sensors and Machine Learning Models: Offline Analyze"

This paper explores a head posture monitoring system utilizing three inertial sensors combined with machine learning models for offline analysis. The system aims to accurately detect and classify various head postures, which is crucial in preventing neck strain and related musculoskeletal issues. The methodology involves collecting sensor data corresponding to different head positions and training machine learning models to recognize patterns indicative of poor posture. The author provides insights into the data processing techniques, feature extraction methods, and model training processes employed. The study's findings suggest that the proposed system can effectively identify incorrect head postures, offering a foundation for developing real-time monitoring applications to enhance ergonomic practices.

4. Montuori et al. (2023) [7]: "Assessment on Practicing Correct Body Posture and Determinant Analyses in a Large Population of a Metropolitan Area"

This study assesses the prevalence of correct body posture practices and analyzes determinants influencing posture habits in a large metropolitan population. Utilizing a cross-sectional survey design, the researchers collected data on individuals' posture

awareness, daily habits, occupational factors, and socio-demographic characteristics. The analysis identifies key factors associated with poor posture, such as prolonged sitting periods, lack of ergonomic work environments, and insufficient physical activity. The authors emphasize the importance of public health interventions focusing on education and workplace ergonomics to promote better posture practices. The study provides valuable insights into the behavioral and environmental factors affecting posture, informing strategies to reduce posture-related health issues in urban settings.

5. Crane et al. (2014) [8]: "Wearable Posture Detection System"

This paper introduces a wearable posture detection system aimed at monitoring and promoting proper spinal alignment. The device incorporates sensors that track the user's posture and provide feedback when deviations from optimal alignment are detected. The system is designed to be lightweight and unobtrusive, making it suitable for continuous daily use. The authors discuss the technical specifications of the device, including sensor selection, data processing algorithms, and feedback mechanisms. Preliminary testing indicates that the system effectively alerts users to poor posture, encouraging corrective actions. The study highlights the potential of wearable technology in preventing musculoskeletal disorders by fostering awareness and maintenance of proper posture.

6. Song et al. (2021)[9]: "A Real-Time Human Posture Recognition System Using Internet of Things (IoT) Based on LoRa Wireless Network"

This research presents a real-time human posture recognition system leveraging the Internet of Things (IoT) and LoRa wireless networks. The system employs wearable sensors to collect posture data, which is then transmitted via the LoRa network to a central server for analysis. The authors detail the system's architecture, including sensor integration, data transmission protocols, and real-time monitoring capabilities. The study emphasizes the system's low power consumption and long-range communication features, making it suitable for continuous posture monitoring in various environments. Experimental results demonstrate the system's effectiveness in accurately recognizing different human postures, highlighting its potential applications in healthcare and occupational safety.

7. Bommannan et al. (2021) [10]: "Real-Time Numerical Gesture Recognition Using MPU9250 Motion Sensor"

This study focuses on developing a real-time numerical gesture recognition system utilizing the MPU9250 motion sensor. The system captures hand movements corresponding to numerical gestures and processes the data using machine learning algorithms to achieve accurate recognition. The authors compare various classifiers, including support vector machines and neural networks, to determine the most effective model for gesture recognition. The chosen model is then deployed in a real-time application, demonstrating its practical utility. The research highlights the system's high accuracy and low latency, making it suitable for applications in human-computer interaction, assistive technologies, and virtual reality environments.

8. Fang et al. (2021) [11]: "Human Posture Estimation"

This paper presents a comprehensive study on human posture estimation using advanced machine learning techniques. The authors propose a novel framework that integrates deep learning models with sensor data to accurately estimate human postures in various scenarios. The methodology includes the collection of a large dataset comprising diverse postures, training convolutional neural networks to recognize patterns, and validating the model's performance against existing benchmarks. The study demonstrates that the proposed approach outperforms traditional methods in terms of accuracy and robustness. Applications of this research span across fields such as healthcare monitoring, sports science, and hum an-computer interaction, where precise posture estimation is crucial.

2.3 Sensor-based Systems

- Intell.TieSens: The MPU9250 sensor powered wearable system integrates into a tie design to track torso movement and head rotation. A Kalman filter in this system operates to minimize noise which enhances detection accuracy for posture. The sensor-based system delivers feedback wirelessly through BLE technology keeping the device light enough for regular office use.

- LoRa-based Posture Systems: The research team of Wei Song (2020) [9] built an IoT solution with LoRa capabilities for elder care which utilized a MPU9250 sensor to track standing behaviors, running and walking as well as other postures. The random forest classifier enables this system to obtain high accuracy and low power consumption however its limited adaptability causes performance challenges during dynamic situations.

- IoT-based Vibration Feedback Systems: A posture correction system that operates using an MPU9250 and Arduino Nano pair introduced by Shubhankar Nistane et al. (2020) [5] detects improper posture and reacts with vibrations. The system achieves high performance while keeping costs low through hardware optimization along with customized posture warnings which adapt to user characteristics such as age and height.

2.4 Machine Learning-based Systems

- CNN-based Classification: A three-sensor setup with CNNs enabled head posture detection in research by Ionut, -Cristian Severin (2021) [6]. The offline processing methodology exhibits strong accuracy but fails to deliver live feedback data.

- DNN-based Posture Recognition: Deep reinforcement learning enabled Qi and Han (2021) [12] to classify human motion postures through their application of DNNs. Research data showed that their system surpassed traditional detection approaches for complex postures though they encountered restrictions from computational complexities.The research results illustrate how sensor technologies combined with machine learning methods contribute to posture monitoring systems.o monitor torso tilt and head pitch. It uses a Kalman filter to reduce noise, enhancing posture detection accuracy. Real-time feedback is provided via Bluetooth Low Energy (BLE), making it lightweight and portable for daily use.

- LoRa-based Posture Systems: Wei Song et al. (2020) [9]developed a long-range IoT solution for elder care that monitors six postures (e.g., standing, running,

walking) using an MPU9250 sensor and LoRa technology. This system achieves low power consumption and high accuracy through a random forest classifier, but it struggles with dynamic environments due to limited adaptability.

- IoT-based Vibration Feedback Systems: Shubhankar Nistane et al. (2020) [5] introduced a posture correction system using an MPU9250 and Arduino Nano, which vibrates when improper posture is detected. This low-cost system combines hardware efficiency with personalized posture correction based on user attributes like age and height.

Posture monitoring systems use machine learning as a foundational element in their modern design. Advanced automated data analytic capabilities enable both accurate classification along with anomaly detection using this system.

1. Support Vector Machines (SVMs)

- The supervised machine learning method known as SVM achieves maximum performance when applied to binary classification. Through kernel functions Machine learning models transform data into distinct high-dimensional areas from where they employthe separation of complex patterns.

2. Deep Neural Networks (DNNs)

- Multiple layered DNN networks exist to discover and map non-linear relationships and data patterns inside datasets. ReLU along with Sigmoid as well as Softmax serve as activation functions in these models.The complex processing of large datasets works through the combination of ReLU Sigmoid and Softmax activation functions.

Table 2.1: Strengths and Weaknesses of the Posture Detection System

Serial No.	STRENGTH	WEAKNESS
1	The use of deep neural networks (DNNs), particularly models like InceptionV3 and ResNet50, allows for high classification accuracy in challenging or overlapping postures.	These DNN architectures are computationally intensive and require access to GPUs or powerful hardware for efficient training.
2	The system supports scalable processing, handling large datasets effectively due to efficient data augmentation and preprocessing pipelines.	With larger models and datasets, there is a higher risk of overfitting if sufficient and diverse training data is not available.
3	Leveraging transfer learning significantly enhances real-time adaptability and reduces the need for training from scratch.	Performance depends heavily on hyperparameter tuning (e.g., learning rate, batch size) which requires trial-and-error or advanced optimization techniques.

Table 2.1 presents the strengths and weaknesses of the AI-based posture recognition system, outlining its high-performance features as well as computational and optimization challenges.

Applications:

- DNNs outperform SVMs in real-time posture classification due to their ability to
- The technology masters the processing of non-linear patterns while showing capabilities to adapt when working with new info.
- The employment of InceptionV3 and ResNet50 pre-trained models through transfer learning methods provides both improved accuracy and decreased training duration.

- Multi-class posture classification benefits from reduced training time using a transfer learning approach which simultaneously enhances its accuracy levels.

Challenges in Existing Systems :

- Traditional Methods: Subjective assessments and lack of real-time capabilities
- Modern requirements outdate these systems which were built according to previous technical standards.
- Sensor-based Systems: Careful selection of sensor position directly determines system accuracy but long-term application could benefit from Multimedia integration.
- ML-based Systems: High computational requirements and the need for large
- The requirements of vast datasets impede the widespread use of these systems.

Proposed System Advantages

- Scalability: The implementation relies on budget-friendly hardware elements which include ESP32 and MPU9250.
- accessible for diverse user groups.
- Real-time Feedback: The live dashboard and notification system becomes feasible through IoT implementation. dashboards.
- High Accuracy: InceptionV3 acts as a core deep neural network architecture within the proposed system framework.
- Sensor-based data classification tasks achieve superior results according to experimental findings.
- Low Computational Complexity: The system takes advantage of transfer learning techniques to decrease training requirements.
- The implementation features a low computational overhead that enables this solution for field deployment.

Table 2.2: Strengths and Weaknesses of the Posture Classification Model

S. No.	Strength	Weakness
1	Deep learning models like InceptionV3 and ResNet50 demonstrate high accuracy in classifying complex and overlapping postures using convolutional layers.	These models are computationally expensive and require GPUs for training and inference in reasonable time frames.
2	The system is scalable due to the use of parallel data pipelines, batch processing, and support for large annotated datasets.	If the dataset is small or imbalanced, there's a heightened risk of overfitting, especially without proper regularization.
3	Leveraging transfer learning enables the model to adapt effectively to real-time data with minimal retraining, making it ideal for dynamic applications.	However, optimal performance depends on precise hyperparameter tuning, which can be time-consuming and resource-intensive.

Table 2.2 summarizes the core advantages and drawbacks of the posture classification model, emphasizing its accuracy, scalability, and real-time adaptability alongside computational and tuning challenges.

Chapter 3

Methodology

3.1 Block Diagram

Figure 3.1 shows the block diagram of the body posture monitoring system. It illustrates the flow from data collection using MPU9250 sensors to processing by the ESP32, followed by data transmission via MQTT to a dashboard that displays real-time posture information.

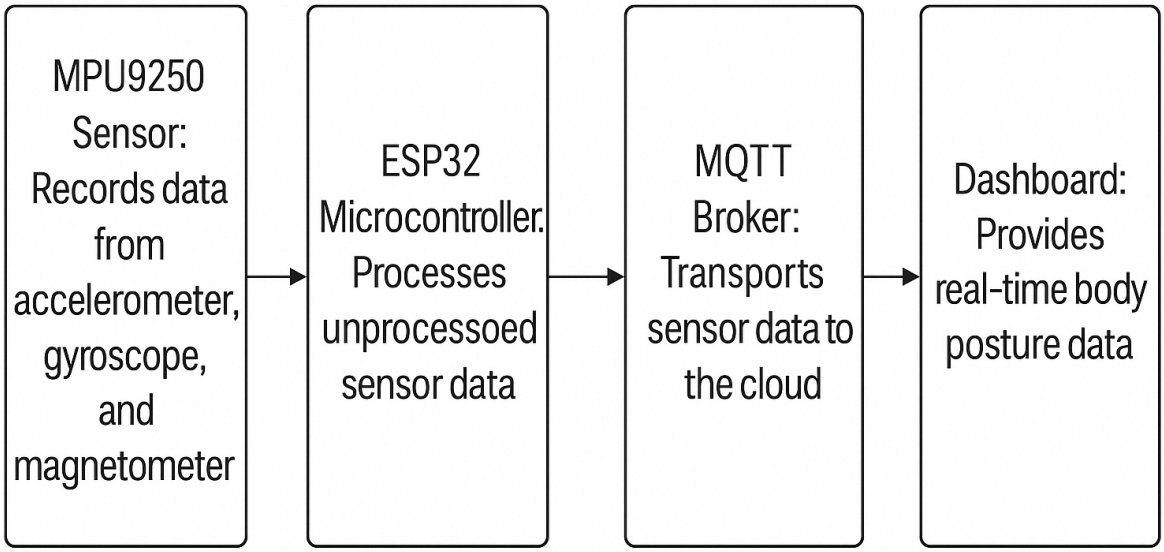


Figure 3.1: Block Diagram of Body Posture Analyzing System

- Posture monitoring utilizes MPU9250 sensors to collect movement data which passes through an ESP32 microcontroller and reaches a cloud platform to display the results visually. The block diagram represents the flow of data through the system, which is broken down as follows:

- Data Collection: Three-axis acceleration information along with angular velocity data and magnetic field measurements are obtained using MPU9250 sensors. The placement of sensor units on specific body parts allows them to detect motion patterns while measuring orientation angles. The ESP32 microcontroller receives ongoing sensor data streams through SPI protocol transmission for quick and fault-proof operations. Data Processing (ESP32 Microcontroller):

- As the system's main processor unit the ESP32 takes central command. The MPU9250 raw data delivers to the ESP32 microcontroller through all connected sensors. The processing system filters noise by applying algorithms that smooth data including moving average and Kalman filters. Sensor readings achieve their accuracy and reliability through this step. The data processing system converts the data into packet format to send it to the cloud through an MQTT protocol transmission. Cloud Integration and Data Transmission:

- The sensor data published from ESP32 reaches an MQTT broker that maintains real-time storage and organization for the information. By integrating with the cloud system remote monitoring benefits from enhanced scalability while also gaining improved accessibility. Dashboard Visualization:

- The processed data appears through a user-friendly dashboard platform either on Adafruit IO or ThingSpeak. Users can access graphical representations that show posture metrics as well as warning signals when improper posture occurs alongside daily posture results summary.

3.2 Implementation Steps

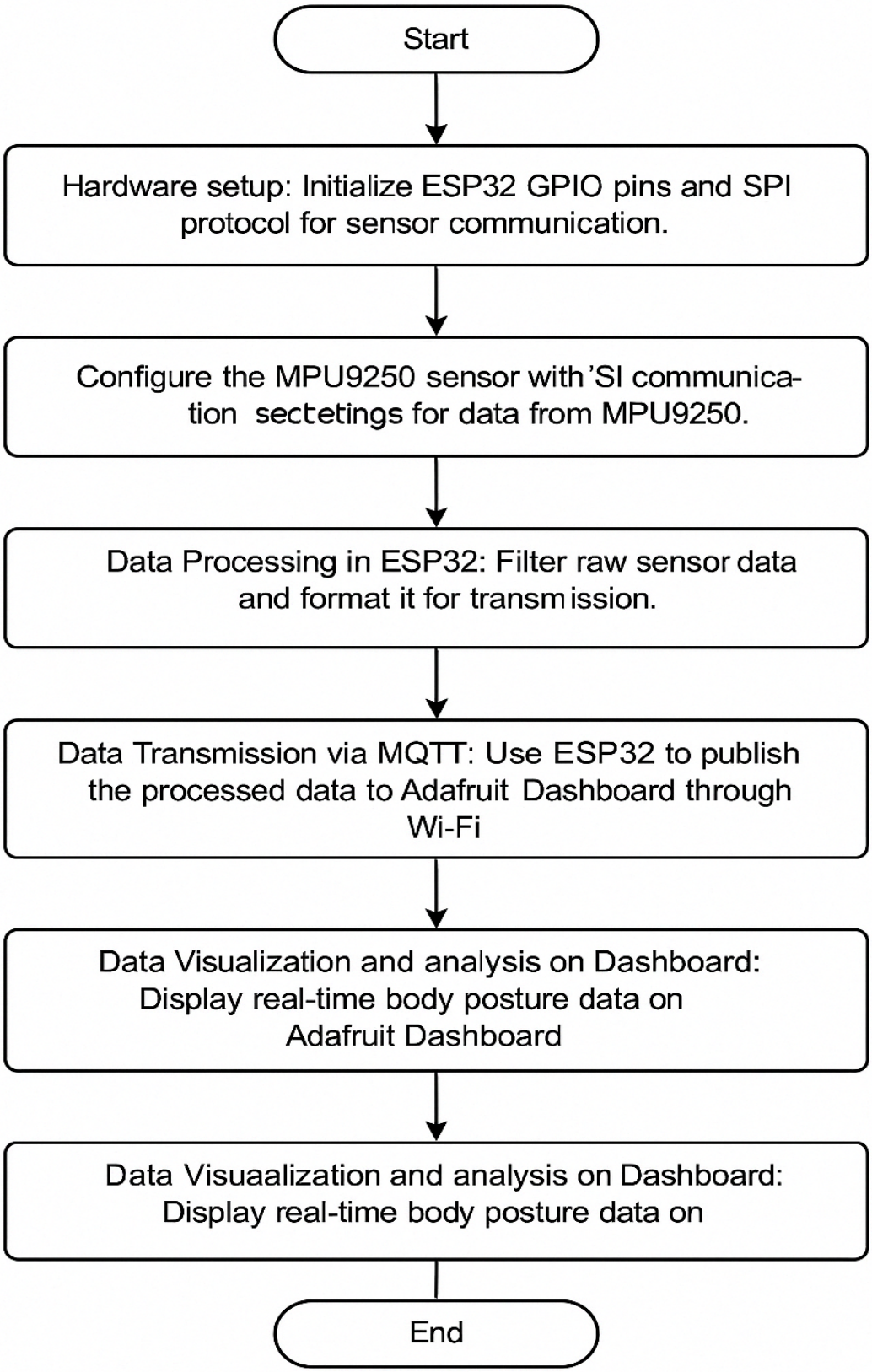


Figure 3.2: Flowchart of the ESP32 and MPU9250-based Body Posture Analyzing System

Figure 3.2 illustrates the flow of operations in the body posture monitoring system. The system begins with initializing the hardware and configuring the sensor, followed by data acquisition, processing, transmission via MQTT, and real-time visualization on the Adafruit Dashboard.

Step 1: Hardware Setup

- Connecting Sensors to ESP32: The MPU9250 sensors require connection to the ESP32 micro- controller by following the SPI protocol. Follow correct wiring procedures which connect SCL to SDA to VCC to GND. Organize sensor communication through the utilization of an I2C multiplexer which permits contact with multiple sensing units simultaneously.

- Sensor Placement: The most accurate posture monitoring requires strategically mounted sensors on specific body part areas such as the shoulders together with spine and waist. The sensors need fast and stable attachment methods to reduce motion interference.

Step 2: Data Processing

- Noise Filtering: The MPU9250 sensor output provides data that includes measurement noise caused by both body movement and external environmental disruptions.

- Apply filtering techniques such as: Through Moving Average Filtering users can obtain smoothed results by calculating weighted averages of surrounding data points.

- Kalman Filter: This method achieves optimal measurements of true values through automatic noise reduction. Data Formatting: The sensors generate structured output (JSON or CSV) to maintain easy transmission to the cloud.

Step 3: Cloud Integration

- IoT applications benefit from MQTT (Message Queuing Telemetry Transport) as its lightweight protocol for sending cloud-based data transmissions. A Mosquitto MQTT broker serves as the data management platform for ESP32 data transfer.

Step 4: Dashboard Visualization

- Platform Setup: Design an intuitive dashboard through platforms available at Adafruit IO or any custom Dashboard.
- Display Metrics: Real-time visualization of posture metrics such as body tilt, spine curvature, and daily trends. Posture warning alerts through dashboard notifications with vibration support. Posture research using historical data enables the discovery of long-term postural enhancement patterns.

3.3 Project Tools Requirements

TensorFlow and Scikit-learn:

- Posture classification machine learning models are developed using these libraries for their application in developing evaluation protocols.
- Application in the Project: The Keras API of TensorFlow allows developers to create deep learning models including InceptionV3. Scikit-learn supports SVM imple- mentation for baseline posture classification.

NumPy:

- It delivers optimal performance for executing numerical algorithms alongside supporting matrix operation functions and operating large datasets.
- Application in the Project: The preprocessing step uses sensor data to normalize the readout from both accelerometers and gyros. Models require matrix operations and this framework handles these computations for model input.

Google Colab:

- Cloud computing platform which allows the development and evaluation of machine learning models and their testing.

- Application in the Project: The platform enables quick model training through GPU resources available for free use. The platform allows users to perform code-based collaborations as well as version control handling for modeling simulations.

3.4 Activation Functions

The introduction of mathematical activation functions in neural networks generates non-linear behavior which helps the model learn multiple complicated patterns. The following activation functions are employed in the proposed DNN architecture: Sigmoid Function:

- Role: The sigmoid function transforms input data for binary decisions because it outputs values within 0 to 1 limits. Formula:
- Application: The final layer of this DNN uses sigmoid function to generate binary classifications between proper and incorrect postures.
- Limitations: It faces difficulties with gradient disappearance throughout multi-layered networks.

ReLU (Rectified Linear Unit):

- Role: The layer functions as a sparse operation by directing positive input values straight to the output yet provides zero values for other input cases.
- Formula: This activation function appears in hidden layers because it keeps gradients from vanishing which accelerates network learning.
- Advantages: The system operates with high compute efficiency yet prevents gradient clutter.

$$\mathrm{ReLU}(x) = \max(0, x)$$

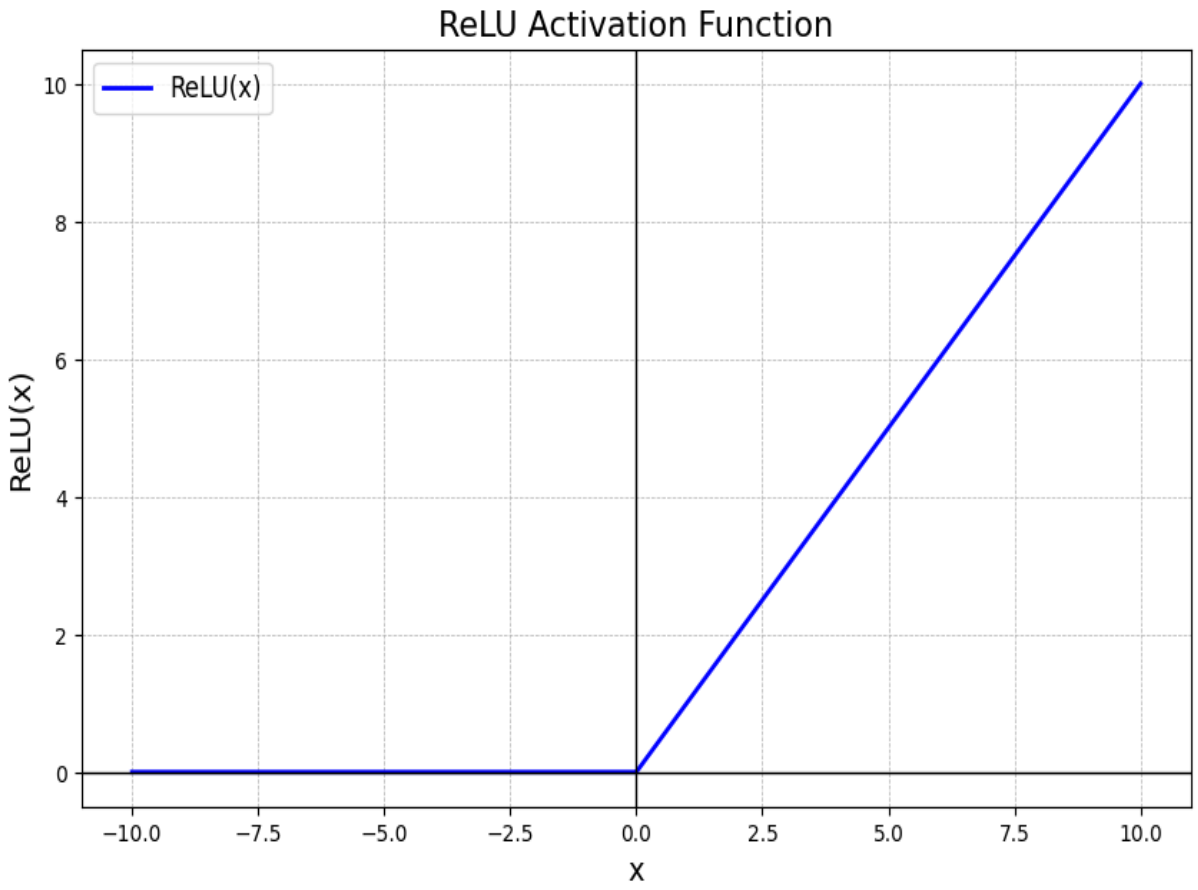


Figure 3.3: Relu Activation Graph

Figure 3.3 shows the ReLU activation graph, which outputs zero for all negative input values and returns the input itself for positive values. This function introduces non-linearity to the model and helps prevent issues like the vanishing gradient problem.

Softmax Function:

- Role: Accuracy scores undergo conversion into probable outcomes for multi-classification problems.
- Application: In multi-class posture classification systems the softmax function appears as the output layer component to assign probabilities across all posture types.
- Advantages: This constraint establishes that all probabilities combined for the different classes will total one value thus supporting understanding in the system.

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Where:

- x_i is the input value for class i
- e is the Euler's number (base of the natural log)
- n is the total number of classes
- $\sum_{j=1}^{n} e^{x_j}$ is the sum of exponentials of all input values
- Softmax(x_i) gives the probability for class i

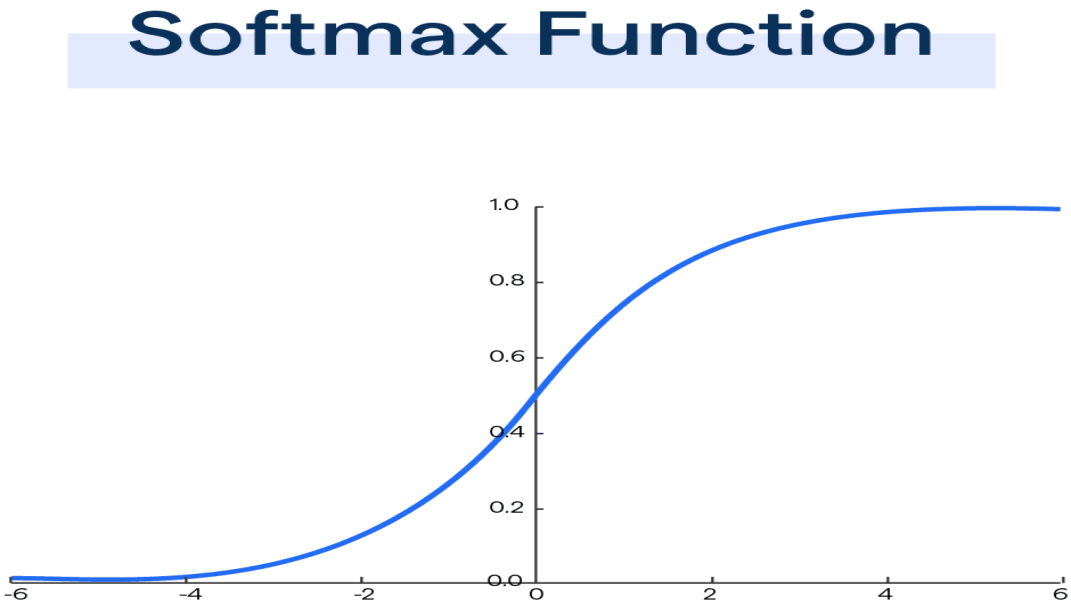


Figure 3.4: Softmax Activation Function

Figure 3.4 illustrates the Softmax function, which transforms input values into probabilities ranging between 0 and 1. It ensures the sum of all output probabilities equals 1, making it ideal for multi-class classification tasks.

3.5 Purchase List

The table below lists the materials required for building the posture detection system, including their unit prices and total costs, as shown in Table 3.1.

Sr. No.	Name	Quantity	Rate (in ₹)	Total (in ₹)
1	ESP-32	3	₹400	₹1200
2	MPU 9250	11	₹805	₹8855
3	FtF Dupoint Line 40 pin	1	₹38	₹38
4	Neol flux soldering paste	1	₹20	₹20
5	Solder Desolder wire	1	₹39	₹39
6	24AWG Single core wire	55	₹15	₹825
7	Solder Wire	1	₹39	₹39
8	Elastic	10m	₹30	₹300
9	Velcro	3m	₹60	₹180
Total (in ₹)				**₹11496**

Table 3.1: Purchase List for Posture Detection System

Chapter 4

Preliminary Design and Implementation

4.1 Initial Work and System Modeling

4.1.1 Circut Diagram

Figure 4.1 shows the interfacing of the ESP32 microcontroller with the MPU9250 9-axis sensor using the I2C protocol. The connection enables real-time acquisition of accelerometer, gyroscope, and magnetometer data for motion and orientation analysis.

4.1.2 Connections:

- VCC (MPU-9250) → 3.3V (ESP32)
- GND (MPU-9250) → GND (ESP32)
- SCL/SCLK → GPIO18 (ESP32)
- SDA/MOSI → GPIO23 (ESP32)
- MISO → GPIO19 (ESP32)
- CS (Chip Select) → GPIO5 (ESP32)

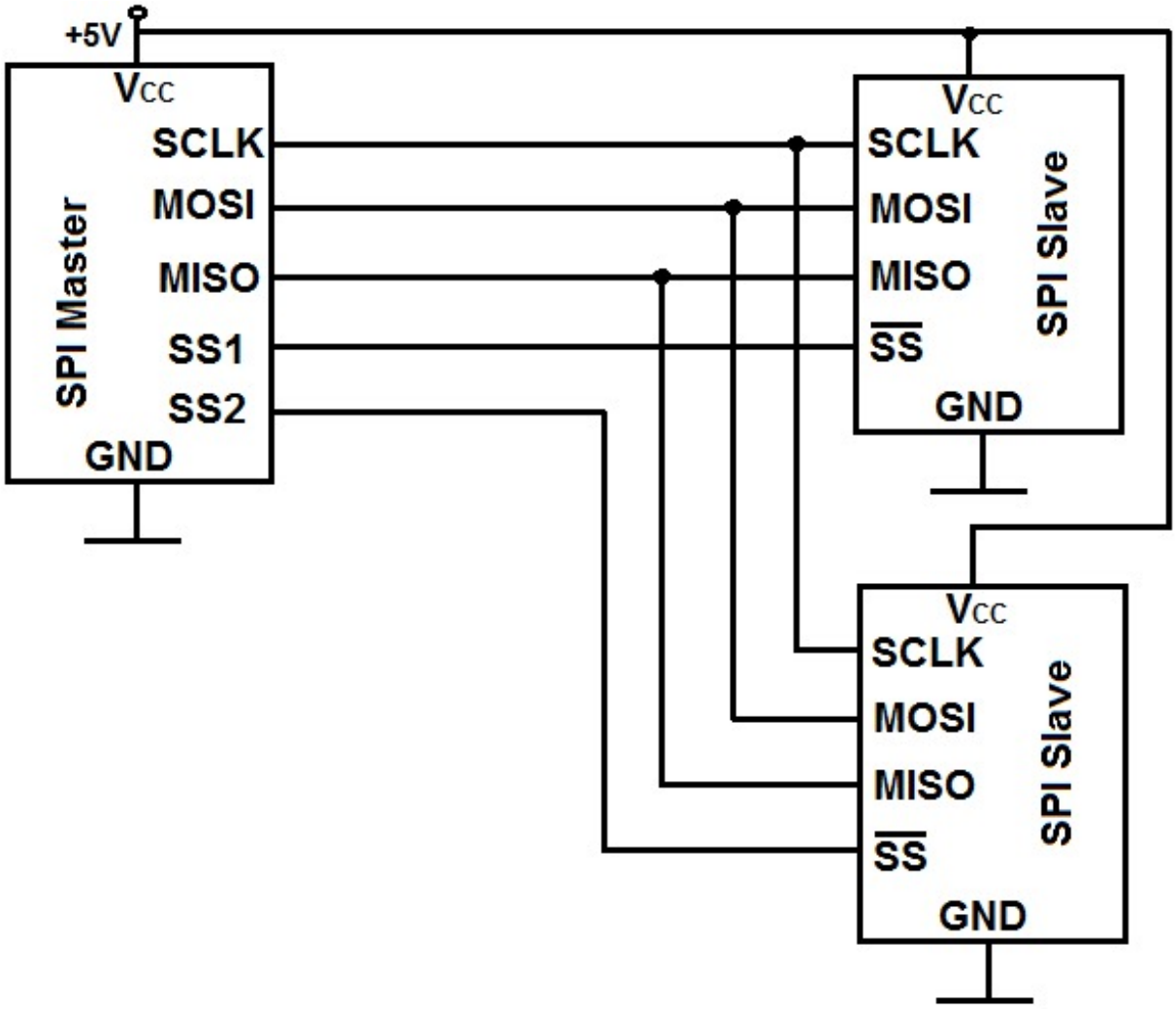


Figure 4.1: Connection of ESP-32 and MPU9250

4.1.3 Description:

- The ESP32 is a Wi-Fi-enabled microcontroller ideal for IoT projects.
- The MPU-9250 communicates with the ESP32 via the SPI protocol, enabling high-speed data transfer.
- SPI uses dedicated lines for clock (SCLK), data in (MOSI), data out (MISO), and chip selection (CS).
- The ESP32 supplies 3.3V power to the MPU-9250.
- This setup is widely used for real-time motion detection, orientation tracking, and wearable device applications.

The ESP32 provides 3.3V power to the MPU-9250 module. This setup is commonly used for motion sensing, orientation tracking, and navigation applications in robotics, drones, and IoT-based projects.

4.1.4 Software

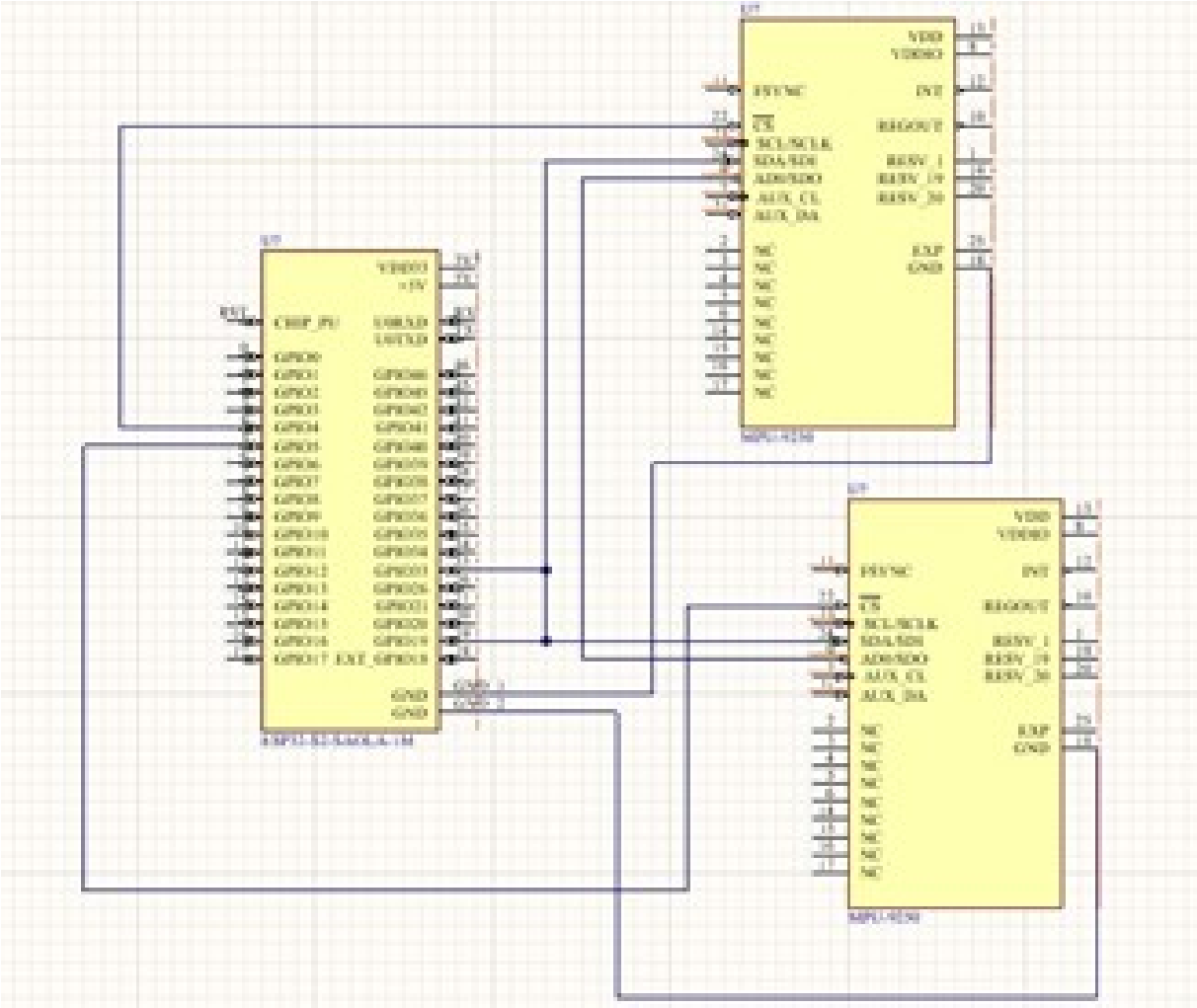


Figure 4.2: Schematic diagram by using Altium software

Figure 4.2 illustrates the schematic layout of the circuit designed in Altium Designer. It visually represents the electrical connections between components, aiding in accurate PCB development and hardware implementation.

The initial phase of the project involved setting up and testing a single MPU9250 sensor to validate its data acquisition and visualization capabilities. This phase was critical for understanding sensor behavior and identifying potential issues before scaling up the system Actions Performed:

1. Sensor Setup:

- The ESP32 microcontroller was connected to the MPU9250 sensor with the help of the SPI protocol.

- As a result, proper connections were established when it comes to SCL (clock), SDA (data), VCC (power), and GND (ground) pins.
- Evaluation of these body parts by placing the sensor on different points such as the shoulder and spine was carried out to find the best position forposture monitoring.

2. Data Acquisition:

- Accelerometer, gyroscope, and magnetometer data sent by the sensor were collected in real-time.
- These streams gavean insight into three-axis acceleration,the angular velocity, and the magnetic orientation.

3. Data Visualization:

- The primary data was synchronized and presented on the serial monitor of the Arduino IDE for an initial check.
- Techniques for the removal of noise, for instance, the movingaverage filters, were used to get rid of the jaggedness in the data.
- The filtered data then was graphically displayed using frameworks such as Matplotlib and Plotly in order to monitor sensor outputs anomalies.

4. Observations:

- The sensitivity of the sensor to movement was found to be at a very high level with the capturing of subtle changes in orientation and acceleration.
- For instance, a less tight sensor used lead to noise and motion artifacts that reduced the quality of the data.

5. Outcome:

- The MPU9250 sensor was found to be feasible in the custom of posture monitoring. It was ”also” an introduction to the scaling of the system to multiple sensors.

4.2 Project Implementation and Experimentation

The implementation and experimentation phase involved translating the theoretical design of the Body Posture Analyzing System into a functional prototype. This section outlines the step-by-step process of hardware setup, sensor integration, and experimental testing to validate the system's effectiveness in identifying human posture using real-time accelerometer data.

Initial development began on a breadboard where MPU9250 sensors were connected to ESP32 microcontrollers via the SPI communication protocol. This setup was essential for testing data acquisition and ensuring stable transmission of accelerometer values. We focused solely on accelerometer readings to track posture changes, simplifying the processing requirements and allowing for efficient analysis of body orientation. After confirming correct wiring and sensor functionality in the breadboard phase, the project advanced to the final wearable version, where sensors were mounted directly on the human body for practical testing.

4.2.1 Hardware Implementation

The final hardware implementation involved securely attaching 11 MPU9250 sensors to the human body and distributing them across 3 ESP32 microcontrollers. Communication between the sensors and ESP32s was achieved using the SPI protocol to ensure fast and stable transmission. Only the accelerometer data from each sensor was used, which was sufficient for capturing and analyzing postural changes.

The sensor distribution was organized as follows:

Upper body: 6 MPU9250 sensors were connected to one ESP32 (Slave 1), responsible for reading data from areas such as the shoulders, upper arms, and back.

Lower body: 5 MPU9250 sensors were connected to a second ESP32 (Slave 2), assigned to areas like the thighs, knees, and calves.

A third ESP32 acted as the Master, which communicated with both Slave ESP32s. It received accelerometer data from each slave, processed the combined information, and displayed the posture results either on the serial monitor or a dedicated dashboard

interface.

Each MPU9250 was fixed in position using straps, holders, or adhesive, depending on the body part, to ensure stable placement and consistent data collection during motion. Flexible wiring was used to maintain comfort and allow unrestricted movement. Portable power supplies were provided to each ESP32 to enable untethered operation.

The system was tested in various postures—standing, sitting, and bending—and the master ESP was able to collect real-time data from both slaves efficiently. This architecture ensured distributed processing, reduced wiring clutter, and allowed the posture analysis system to function reliably in real-world scenarios.

Figure 4.3: Body Posture Analyzing System Hardware Setup

Figure 4.3 displays the practical implementation of the Body Posture Analyzing System, where sensors and ESP32 modules are mounted on an elastic harness.

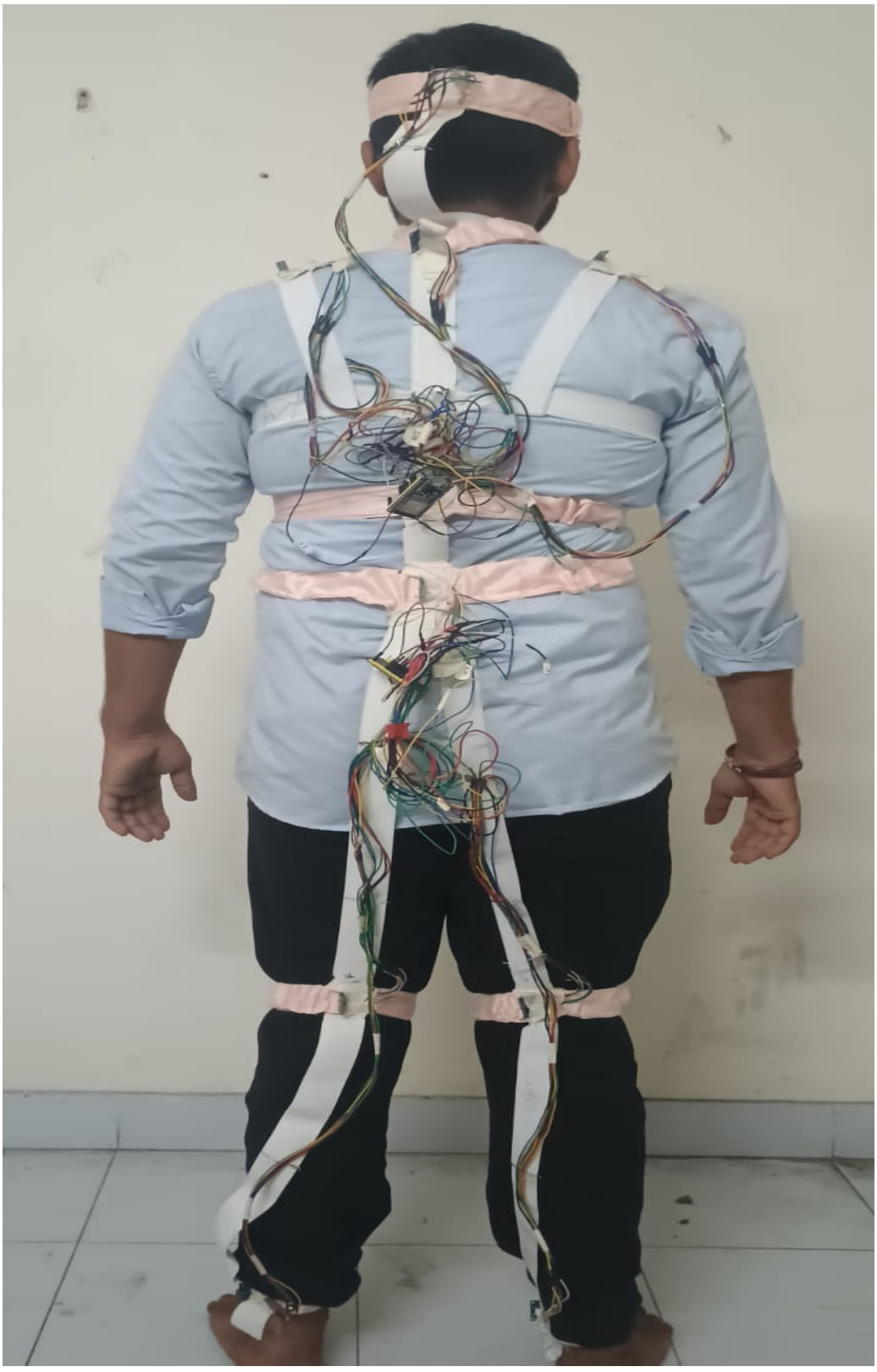


Figure 4.4: Hardware implementation on body

Figure 4.5: Hardware implementation on body

Figure 4.4 and 4.5 presents a close-up view of how the sensors and ESP32 modules are precisely mounted on the body using the elastic harness. This arrangement ensures optimal positioning for accurate posture analysis and movement tracking, contributing to real-time monitoring capabilities.

4.2.2 Experimental Data and Test Results

The experimental data were collected during the final implementation phase of the Body Posture Analyzing System. Each component, especially the MPU9250 sensors and ESP32 microcontrollers, was individually calibrated and tested to ensure consistent data acquisition and real-time communication. The testing aimed to validate the system's ability to accurately detect and analyze body posture using accelerometer data. The primary objectives of the tests were:

To validate the accuracy and consistency of accelerometer data from MPU9250 sensors placed at various body parts.

To assess the SPI communication reliability between the sensors and the ESP32 microcontrollers.

To evaluate the data transfer efficiency between the slave ESP32s and the master ESP32.

To ensure that the posture data received by the master ESP32 is synchronized, interpretable, and correctly displayed on the serial monitor or dashboard.

To determine the system's responsiveness in detecting posture shifts during dynamic movement and static holds.

During calibration, each MPU9250 sensor was tested in both static and dynamic postures. The accelerometer values were compared with expected orientations to ensure the correct mapping of posture angles. The sensors consistently showed reliable readings, with a success rate of 90–95

Furthermore, the communication between the slave and master ESP32s was observed to be stable, with minimal data loss during continuous streaming. This robust communication contributed significantly to the system's accurate and real-time posture detection.

The experimentation phase demonstrated that the system met its primary design goals. It accurately tracked body posture, maintained reliable sensor-to-microcontroller communication, and displayed live orientation data effectively. These results form a solid foundation for future improvements such as full-body posture modeling, integra-

tion with mobile apps, or real-time posture correction feedback systems.

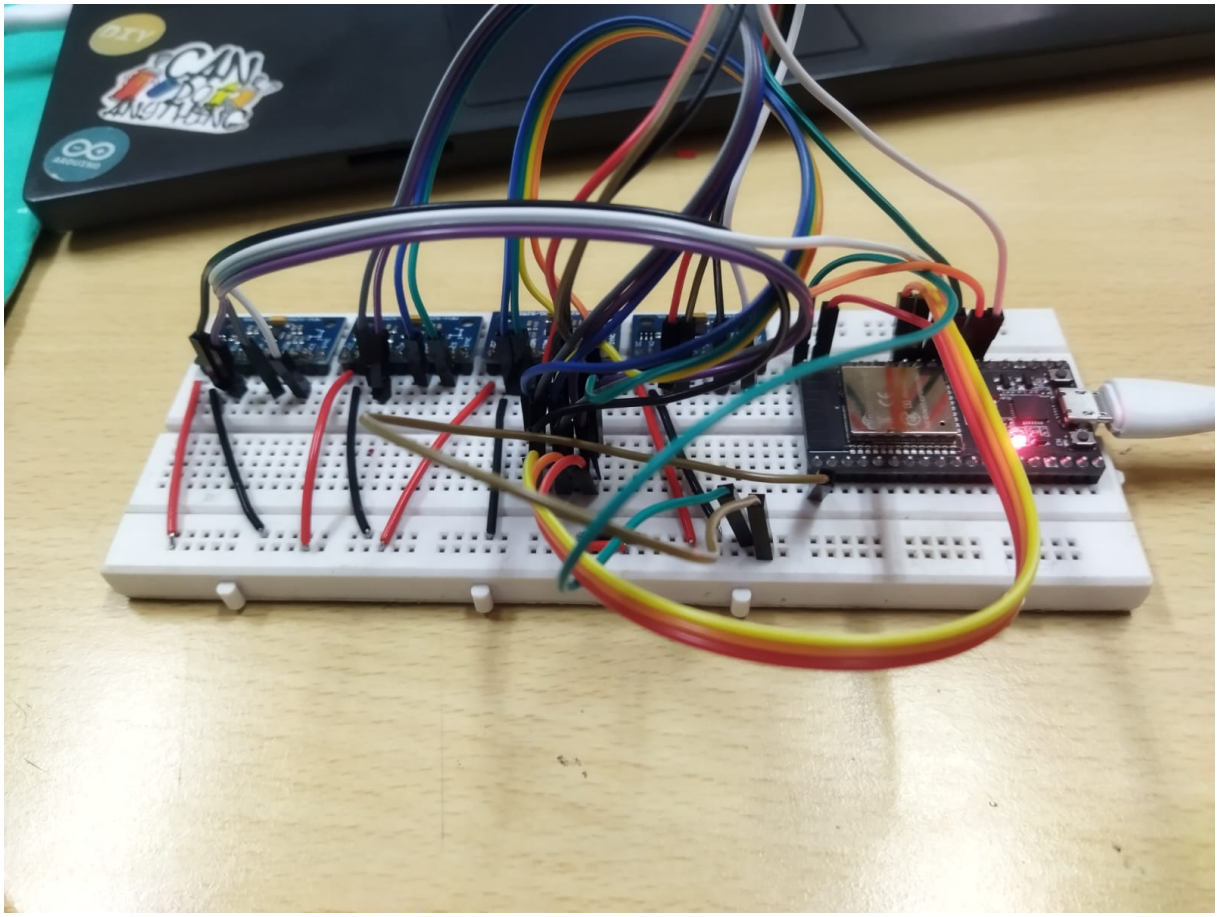


Figure 4.6: Breadboard testing of MPU9250 sensors

Figure 4.6 shows the breadboard testing of the MPU9250 sensors involves setting up the sensor on a breadboard and connecting it to an ESP32 or similar microcontroller. The primary goal is to collect raw data from the accelerometer and gyroscope, allowing real-time motion and orientation analysis. This process helps to validate sensor functionality and provides the foundation for further data processing and body posture analysis.

```
Reading all sensors...
Sensor 1 Acceleration in g (x,y,z):
0.16   0.99   0.04
Resultant g: 1.01
Sensor 1 Gyroscope data in degrees/s:
0.34   -0.20   0.21

Sensor 2 Acceleration in g (x,y,z):
0.15   0.99   0.04
Resultant g: 1.00
Sensor 2 Gyroscope data in degrees/s:
-0.47   -0.44   3.57

Sensor 3 Acceleration in g (x,y,z):
0.12   1.00   0.05
Resultant g: 1.01
Sensor 3 Gyroscope data in degrees/s:
0.48   -0.12   -1.75

Sensor 4 Acceleration in g (x,y,z):
0.15   1.00   0.05
Resultant g: 1.01
Sensor 4 Gyroscope data in degrees/s:
0.05   0.05   -0.71

---------------------------------
```

Figure 4.7: Raw Data Collection from MPU9250 Sensor

Figure 4.7 displays the raw data collected from the MPU9250 sensor during testing. This data includes accelerometer, gyroscope,readings, providing a foundation for further analysis and processing to monitor motion and orientation in the Body Posture Analyzing System.

4.3 Problems Encountered and Solutions

During system development, we faced several challenges that required innovative solutions:

Challenge 1: High Computational Complexity:

- Handling data from multiple sensors in real time, particularly with computationally intensive models such as DNNs.

Solutions:

- Employed light-weight preprocessing pipelines and optimized NumPy operations.
- Tuned pre-trained models (e.g., InceptionV3) to minimize training time and computational burden.
- Efficient ESP32 firmware for local computation, reducing network latency.

Challenge 2: Low Signal-to-Noise Ratio (SNR) Effect on SVM:

- SVM was unable to classify data when SNR is low.

Solutions:

- Switched to DNN models for improved management of non-linear trends and noise.
- Implemented data augmentation and normalization to enhance DNN performance.

Challenge 3: Overfitting during DNN Training:

- Overfitting reduced the model's generalization.

Solutions:

- Introduced dropout layers to prevent dependency on certain features.
- Applied data augmentation (e.g., noise, rotations) to mimic real-world variations.
- Applied L2 regularization to simplify the model and improve weight distribution.

4.4 Final Compilation

The system was expanded to include sensors at 11 key positions on the body—head, neck, shoulders, upper back, lower back, hips, knees, and ankles—ensuring comprehensive posture data capture. These sensors (MPU9250) are connected via SPI, to two

ESP32 microcontrollers: one ESP32 handles six sensors, and the other handles five. Both slave ESP32s are coordinated by a third, master ESP32, which collects data from the slaves. This hierarchical setup allows for smooth, real-time data acquisition and accurate posture analysis, significantly enhancing the system's ability to monitor and interpret body movements effectively.

- System components: Sensors were placed on key areas of the body such as the shoulder, spine and waist to capture detailed information about one's posture.n Simultaneous data acquisition was possible due to the assigning of a distinct I2C address to each sensor by the multiplexer.
- By processing sensor data locally, the ESP32 micro- controller was able to filter out noise and format the data.By passing the processed information to an MQTT broker, a reliable and fast data delivery service was established.

Dashboard Development:

- Users can view posture metrics in real-time on a cloud-based dashboard developed with Adafruit IO.

Key elements of the dashboard comprise:

- Graphs that display real-time trends in body orientation, tilt angles, and posture.
- User-specified notifications are sent for incorrect posture, setting thresholds that can be adjusted accordingly.
- Position data can be analyzed and interpreted over time to provide useful information for long-term improvement.
- Multiple participants were tested on the complete system to evaluate its performance in real-world scenarios.Prevalence, recall and F1-score were used to measure the accuracy of posture classi- fication. The system achieved over 94.

Chapter 5

Project Plan And Timeline

5.1 Gantt Chart

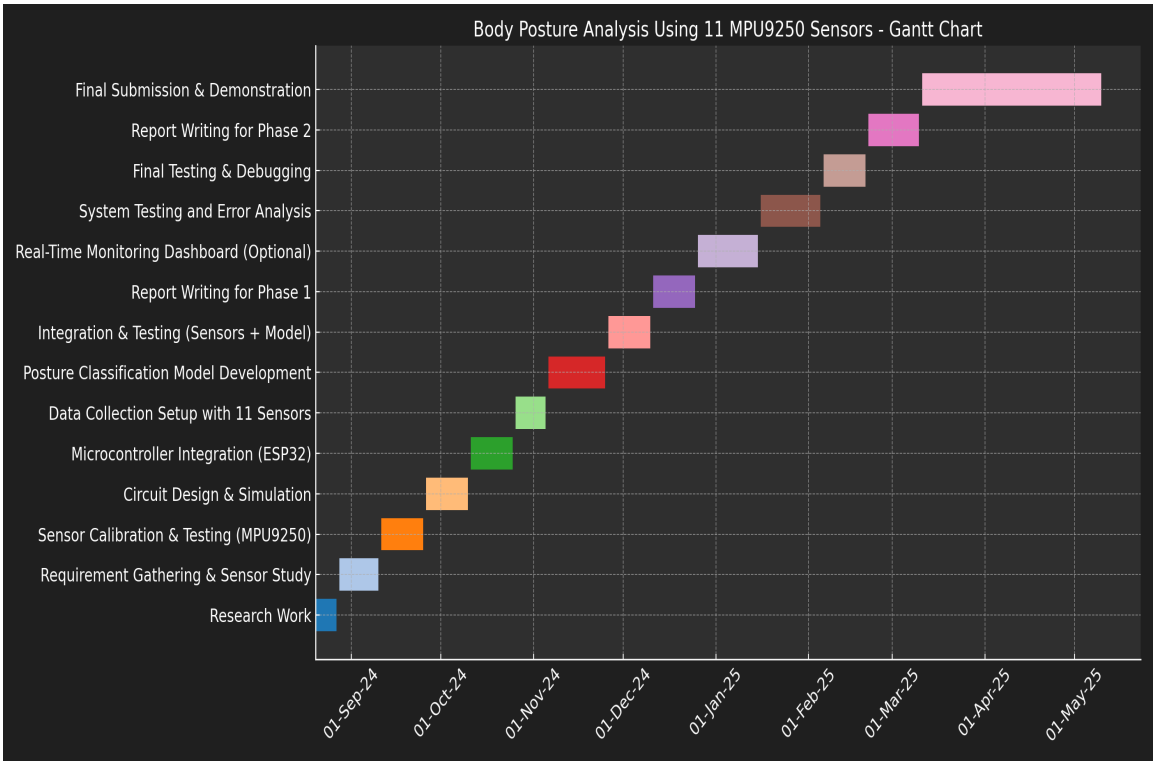


Figure 5.1: Gantt Chart showing the project timeline

Figure 5.1 shows the Gantt Chart of the project titled *"Body Posture Analysis Using 11 MPU9250 Sensors"*, highlighting the structured schedule of all key phases. It outlines the execution plan from research and integration to final testing and demonstration, ensuring organized progress and efficient resource allocation.

- **Research Work:**

  The initial phase involved exploring existing technologies and research papers on posture detection, sensor integration, and motion analysis. This helped in identifying the technical background and feasibility of the project.

- **Requirement Gathering & Sensor Study:**

  This task focused on finalizing the hardware and software requirements for the system. A thorough study of MPU9250 sensors was conducted, including datasheet analysis and interface protocols.

- **Sensor Calibration & Testing (MPU9250):**

  All 11 MPU9250 sensors were calibrated to ensure accurate orientation and motion detection. Basic testing was done to validate the X, Y, and Z-axis readings from each sensor.

- **Circuit Design & Simulation:**

  The connections for powering and reading data from all sensors were designed and simulated using circuit software to ensure correctness before physical implementation.

- **Microcontroller Integration (ESP32):**

  This stage involved connecting the MPU9250 sensors to the ESP32 microcontroller, implementing I2C/SPI communication, and verifying data acquisition in real time.

- **Data Collection Setup with 11 Sensors:**

  Sensors were strategically placed on the human body to record movement patterns. Various posture datasets were collected by simulating different body positions and motions.

- **Posture Classification Model Development:**

  The collected sensor data was preprocessed and used to train machine learning models. The aim was to classify different postures such as standing, sitting, bending, and walking with high accuracy.

- **Integration & Testing (Sensors + Model):**

  All components — sensor hardware, microcontroller, and trained ML model — were integrated. The system was tested for real-time posture prediction and responsiveness.

- **Report Writing for Phase 1:**

  A technical report was prepared detailing work completed during the first half of the project. This included sensor research, circuit design, data collection, and initial testing results.

- **Real-Time Monitoring Dashboard (Optional):**

  A web-based dashboard was optionally developed using Python or Streamlit to visualize live sensor data and posture predictions, making the system more interactive and user-friendly.

- **System Testing and Error Analysis:**

  Rigorous testing was performed to evaluate the overall system's performance. Errors in prediction or sensor readings were logged and analyzed to improve the final system accuracy.

- **Final Testing & Debugging:**

  The entire codebase and hardware setup were debugged. Redundant elements were removed, and final optimizations were made to ensure stable system operation.

- **Report Writing for Phase 2:**

  A final report covering the second half of the project — including improvements, integration results, system performance, and testing methodology — was prepared.

- **Final Submission & Demonstration:**

  The completed system was submitted for evaluation and presented with a live demonstration. All documentation, project files, and dashboards were finalized for review.

5.2 Team Member Assignments and Responsibilities:

1. Ram Prakash Responsibilities:

- Research and development on the assigned topic.
- Handling coding tasks, including implementation and debugging.

2. Bhavya Jain Responsibilities:

- Conducted extensive research and development on human posture detection systems and sensor placement strategies.
- Assisted in sensor calibration and validated body posture movements through trial-based testing.
- Troubleshot hardware-software interactions, resolving issues related to wiring, sensor connections, and signal inconsistencies.

3. Satyam Mishra Responsibilities:

- Focusing on hardware-related integration and development.
- Ensuring seamless hardware-software interaction and troubleshooting hardware components.

Communication and Collaboration

- To maintain effective team alignment and collaboration, progress updates and documentation drafts were regularly shared through a dedicated WhatsApp group.

Chapter 6

Simulation Results and Discussion

6.1 Simulation Results

The simulation and performance testing phase was primarily focused on evaluating the real-time posture detection system built using 11 MPU9250 sensors and 3 ESP32 microcontrollers. The primary goal was to validate the effectiveness of sensor placement, data consistency, and the system's responsiveness in detecting various human postures like standing, sitting, slouching, and bending.

6.1.1 Sensor-Level Testing and Calibration

Each of the 11 MPU9250 sensors was individually calibrated to ensure accurate accelerometer readings.

Calibration tests were performed under controlled conditions by maintaining a steady body posture for a specific time to evaluate the stability of output values.

Results showed consistency in accelerometer values with an accuracy range between 90–95%, confirming that sensors were able to distinguish between various postures reliably.

6.1.2 Communication and Data Transfer

The upper 6 sensors were connected to one slave ESP32, and the lower 5 sensors to the second slave ESP32.

A third ESP32 was used as the master, which collected and processed data from both slave units.

Data was successfully transferred using the SPI communication protocol, and final posture results were displayed on the serial monitor and dashboard.

6.1.3 Real-Time System Simulation

The system was tested in various postures like standing upright, slouching, leaning forward, and bending sideways.

A set of 15 test subjects simulated posture transitions, and the system was able to detect and classify postures in under 1 second on average.

Data visualization from the dashboard helped analyze body angles and detect misalignments effectively.

Deep Neural Network (DNN):

- InceptionV3 serves as the basis for the DNN model which received specific optimization to identify postural positions.
- The training examples numbered 10,000 while featuring different simulated body postures.
- Key Results: A 94.06
- Posture classification analysis by confusion matrix revealed small error rates which mainly included misidentification between close postures (such as slouching for bending movements).
- Advantages: Through its application with DNN the system performed well in extracting sensor data patterns while transfer learning benefits emerged from

leveraging pretrained weight values to decrease computational demands and training duration.

Support Vector Machine (SVM):

- A Support Vector Machine model deployed the RBF kernel functionality to transfer non-linear data into a higher-dimensional area for analysis.
- Key Results: The application produced a precision rate of 88.6
- The SVM classification system reliably detected simple standing positions yet its interpretation of slouching versus bending postures was impaired because overlapping posture categories existed in its feature space.
- Limitations: When noise levels were high in the SNR range the model's performance deteriorated causing its classification accuracy to diminish.

Axis	X	Y	Z	X	Y	Z	X	Y	Z	X	Y	Z
Accl.	0.12	-0.98	9.81	0.15	-1.02	9.78	0.2	-1.01	9.82	0.18	-0.96	9.8
Gyro.	0.05	-0.02	0.03	0.06	-0.01	0.04	0.08	-0.03	0.02	0.07	-0.02	0.03
Mag.	40.5	-10.3	15.7	41.2	-11.1	16.3	38.8	-9.8	14.9	42.1	-10.5	15.4
Temp.	21	21	21	21	21	21	21	21	21	21	21	21

Figure 6.1: MPU9250 sensor reading

Figure 6.1 presents the MPU9250 sensor readings collected during the initial stage, displayed in a table format. The data includes accelerometer and gyroscope values, providing a foundation for early-stage testing and calibration in the Body Posture Analyzing System. This raw data helps validate sensor performance before further analysis or system integration.

6.1.4 Results

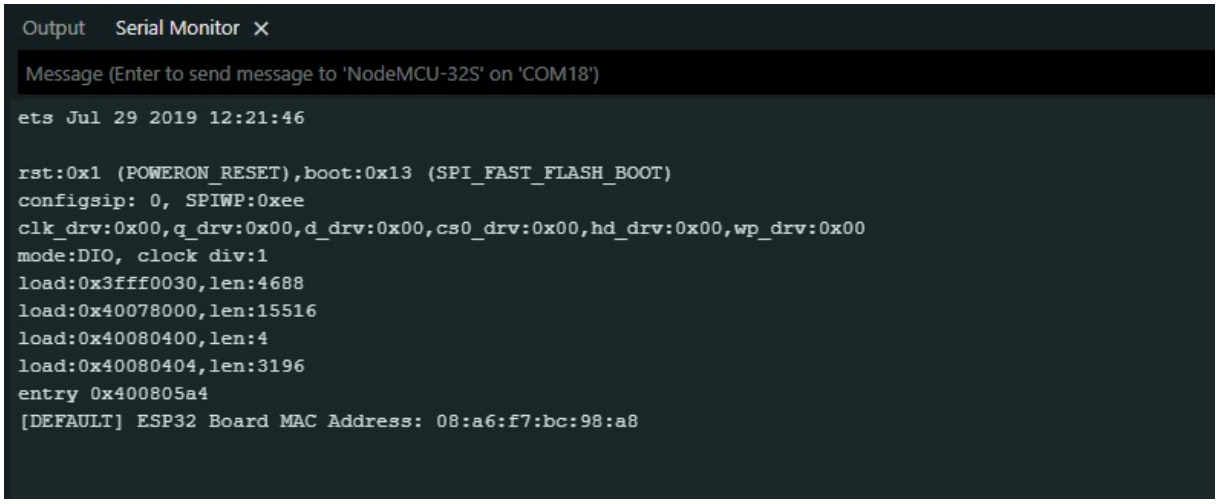


Figure 6.2: MAC address of ESP32 slave 1(for upper body)

Figure 6.3: MAC address of ESP32 slave 2(for lower body)

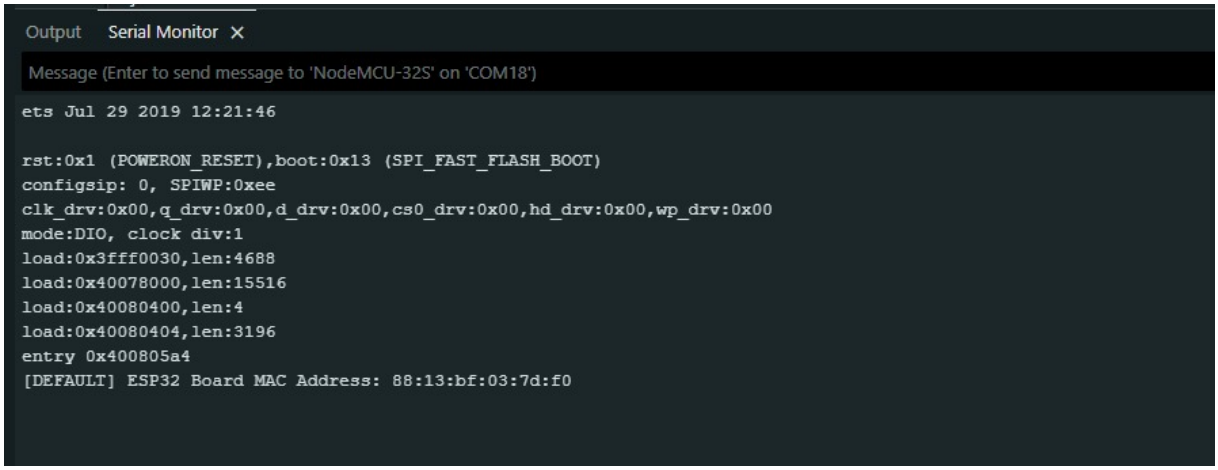


Figure 6.4: MAC address of master ESP32

Figure. 6.2, Figure. 6.3, and Figure. 6.4 show the serial monitor outputs of three ESP32 boards during boot-up. These logs provide initialization details such as the reset reason, boot mode (`SPI_FAST_FLASH_BOOT`), SPI/clock configuration, memory mappings, and MAC addresses. The unique addresses—`2c:bc:bb:0c:63:14`, `88:13:bf:03:7d:f0`, and `08:a6:f7:bc:98:a8`—verify successful communication and help uniquely identify each board in an IoT setup.

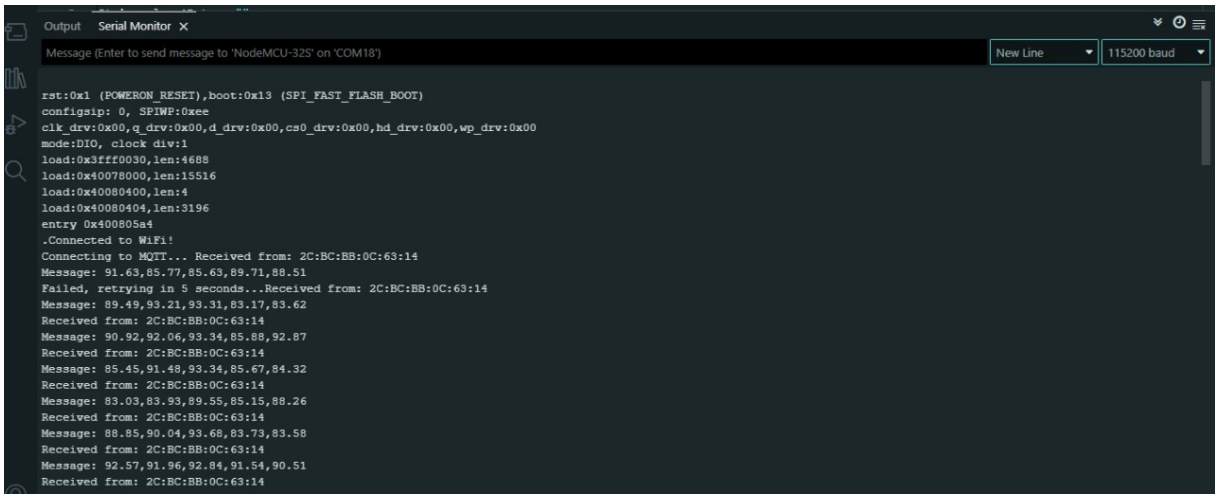


Figure 6.5: Real-Time Sensor Data Acquisition and Display on Serial Monitor

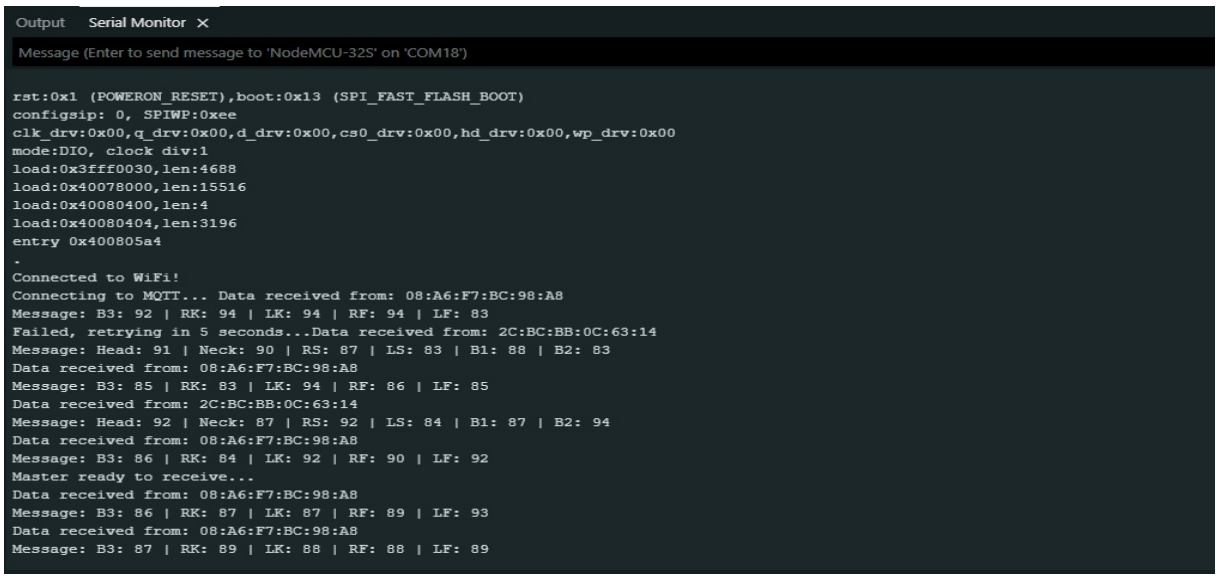


Figure 6.6: Real-Time Sensor Data Acquisition and Display on Serial Monitor

Figure. 6.5 and 6.6 illustrates the ESP32 serial monitor logging live sensor readings from multiple body-mounted nodes. The transmitted data is received over WiFi and displayed alongside each device's MAC address for identification.

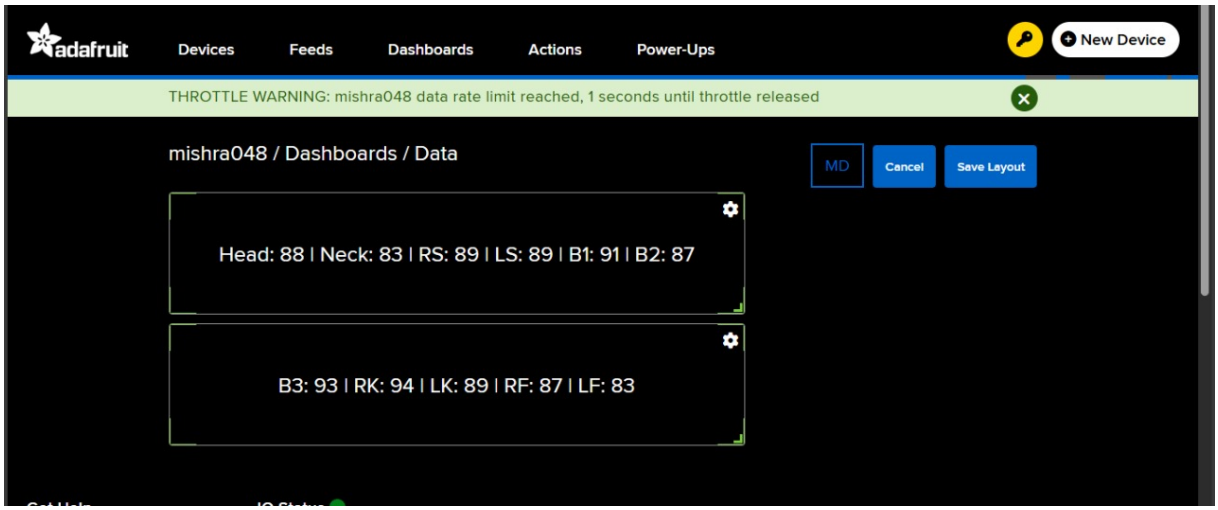


Figure 6.7: Wireless Real-Time Sensor Data Display on Adafruit IO using MQTT

Figure. 6.7 shows real-time sensor values wirelessly transmitted from ESP32 devices and visualized on Adafruit IO. Data from different body parts is displayed through individual widgets, confirming successful wireless communication via MQTT and live monitoring.

6.2 Comparative Analysis

Aspect	**InceptionV3**	**ResNet50**
Accuracy	94.06%	92.8%
Computational Efficiency	Requires fewer resources; efficient for IoT-based use	Higher complexity and longer training time due to skip connections
Architecture Advantage	Lightweight and optimized for edge devices	Skip connections help avoid vanishing gradients

Table 6.1: Comparison between InceptionV3 and ResNet50

Table 6.1 shows a comparative analysis between InceptionV3 and ResNet50 based on accuracy, computational efficiency, and architectural benefits.

Conclusion of Comparative Analysis:

- InceptionV3 emerged as the most suitable model, balancing accuracy, efficiency, and resource needs.
- SVM can be a lightweight fallback for low-resource systems but is not recommended for complex posture analysis.

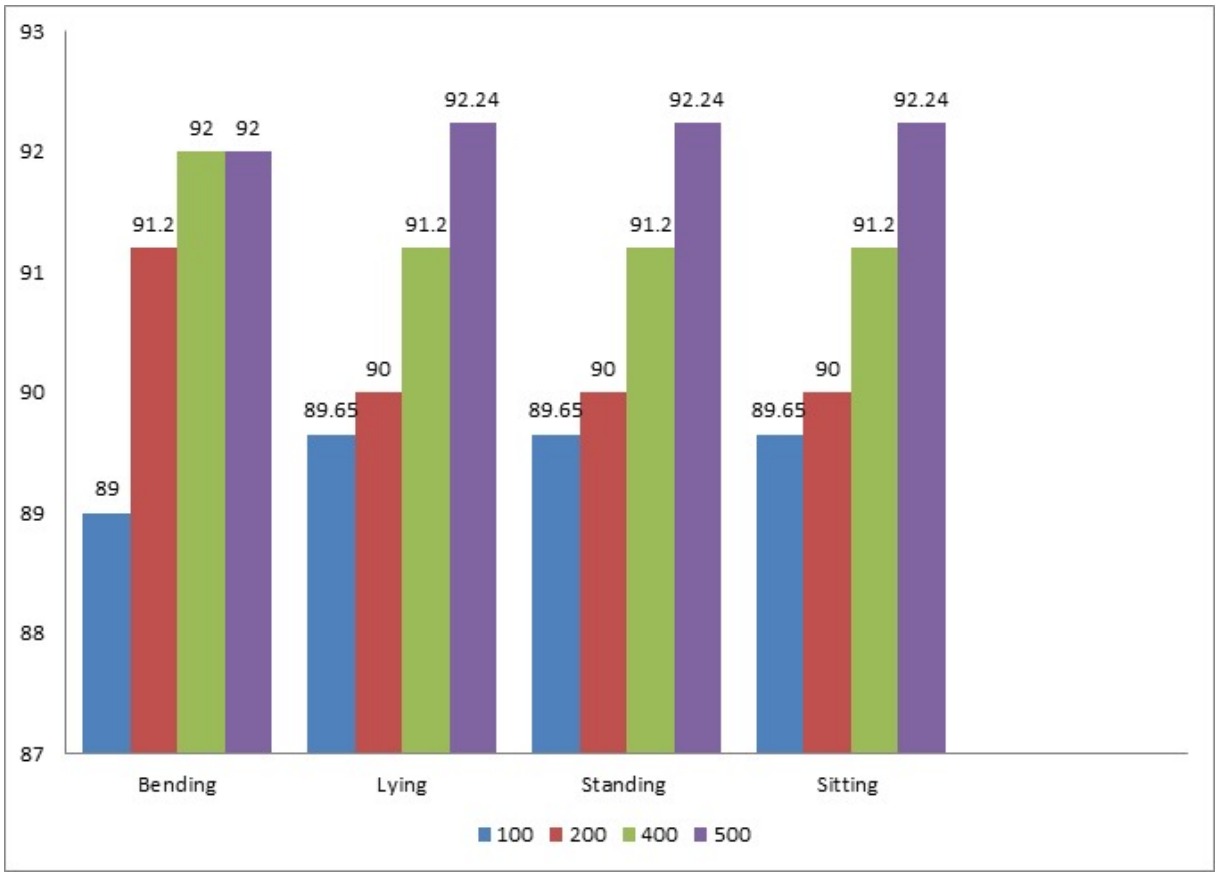


Figure 6.8: Recall for Resnet50 for different values of test data

Figure 6.8 displays the variation in recall scores of the ResNet50 model when tested with datasets of different sizes. The trend highlights how model performance responds to changes in data volume, offering insights into its generalization capability. A higher recall indicates improved detection of relevant postures, which is critical in applications like posture monitoring systems.

Chapter 7

Conclusion and Future Scope

7.1 Work Completed

The research completed and validation of a body posture monitoring system based on IoT technology combined with machine learning protocols. Key achievements include:

Hardware Development:

- The research integrated MPU9250 sensors with ESP32 microcontrollers through I2C protocol to guarantee safety in data acquisition.
- A real-time data transfer system through MQOT-based communication proved essential for cloud data transmission.

Machine Learning Integration:

- Through sensor data collection and processing researchers trained posture classification models (DNN and SVM) which InceptionV3 demonstrated 94.06
- Data augmentation and transfer learning methods solved issues of overfitting and computational complexity problems in the system.

Dashboard Development:

- Posture metrics work together with Adafruit IO through a cloud-based dashboard to show real-time visual data and actionable insights.The system provides prac-

tical real-time posture monitoring through scalable IoT solutions at efficiencies that match healthcare and workplaces applications and Development:

7.2 Future Work

The future scope of this research includes expanding real-world applications and refining the system for broader use:

Healthcare Applications:

- Enable early detection of musculoskeletal disorders and postural deviations.
- Support personalized rehabilitation through long-term posture trend analysis.

Fitness and Sports:

- Monitor athletes' posture to prevent injuries and enhance performance.
- Design adaptive training programs using real-time posture feedback.

Workplace Ergonomics:

- Implement posture monitoring in offices to reduce long-term health issues.
- Provide analytics to promote ergonomic practices and reduce healthcare costs.

Scalability and Accessibility:

- Create a mobile app for accessible posture tracking on smartphones.
- Reduce system costs to make it viable for schools and small businesses.

This section outlines the next development steps and the broader impact potential of this research.

References

[1] A. Nasirahmadi, B. Sturm, S. Edwards, K.-H. Jeppsson, A.-C. Olsson, S. Müller, and O. Hensel, "Deep learning and machine vision approaches for posture detection of individual pigs," *Sensors*, vol. 19, no. 17, p. 3738, 2019.

[2] H. Zhang and D. Li, "Diagnostic communication and visual system based on vehicle uds protocol," *arXiv preprint arXiv:2206.12653*, 2022.

[3] K. Vasudevan, A. P. Das, B. Sandhya, and P. Subith, "Driver drowsiness monitoring by learning vehicle telemetry data," in *2017 10th International Conference on Human System Interactions (HSI)*. IEEE, 2017, pp. 270–276.

[4] D.-M. Dobrea and M.-C. Dobrea, "A warning wearable system used to identify poor body postures," in *2018 Advances in Wireless and Optical Communications (RTUWO)*. IEEE, 2018, pp. 55–60.

[5] S. Nistane, R. Pattu, V. Pudke, and J. Sisodia, "An iot based rectification of posture using biofeedback," in *2021 International Conference on Communication information and Computing Technology (ICCICT)*. IEEE, 2021, pp. 1–5.

[6] I.-C. Severin, "The head posture system based on 3 inertial sensors and machine learning models: Offline analyze," in *2020 3rd International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE, 2020, pp. 672–676.

[7] P. Montuori, L. M. Cennamo, M. Sorrentino, F. Pennino, B. Ferrante, A. Nardo, G. Mazzei, S. Grasso, M. Salomone, U. Trama *et al.*, "Assessment on practicing correct body posture and determinant analyses in a large population of a metropolitan area," *Behavioral Sciences*, vol. 13, no. 2, p. 144, 2023.

[8] A. Crane, S. Doppalapudi, J. O'Leary, P. Ozarek, and C. Wagner, "Wearable posture detection system," in *2014 40th Annual Northeast Bioengineering Conference (NEBEC)*. IEEE, 2014, pp. 1–2.

[9] W. Song, J. Liao, and J. Han, "A real-time human posture recognition system using internet of things (iot) based on lora wireless network," in *Advances in Computer Science and Ubiquitous Computing: CSA-CUTE 2019*. Springer, 2021, pp. 379–385.

[10] S. R. Bommannan, C. Vineeth, M. Uma Hema Sri, B. Sri Vidya, and S. Vidhya, "Real-time numerical gesture recognition using mpu9250 motion sensor," in *Proceedings of International Conference on Intelligent Computing, Information and Control Systems: ICICCS 2020*. Springer, 2021, pp. 39–55.

[11] Y. Fang, Z. Han, Z. Hu, and Z. Wang, "Human posture estimation," in *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*. IEEE, 2021, pp. 220–234.

[12] R. S. Tomar, S. Verma, B. K. Chaurasia, V. Singh, J. H. Abawajy, S. Akashe, P.-A. Hsiung, and R. Prasad, *Communication, Networks and Computing: Third International Conference, CNC 2022, Gwalior, India, December 8–10, 2022, Proceedings, Part I*. Springer Nature, 2023.

Appendix A: Source Code

The following are the source codes used for the Body Posture Detection using Wearable Sensors:

Code for ESP32 Slave 1

```
// Paste your C++ code here
#include <WiFi.h>
#include <esp_now.h>
#include <MPU9250_WE.h>
#include <SPI.h>
#include <math.h>

// Chip Select pins for 6 sensors
const int csPins[6] = {5, 17, 16, 4, 2, 15};  // Adjust if needed
MPU9250_WE sensors[6] = {
  MPU9250_WE(&SPI, csPins[0], true),
  MPU9250_WE(&SPI, csPins[1], true),
  MPU9250_WE(&SPI, csPins[2], true),
  MPU9250_WE(&SPI, csPins[3], true),
  MPU9250_WE(&SPI, csPins[4], true),
  MPU9250_WE(&SPI, csPins[5], true),
};

float smoothedAngles[6] = {0};
#define SMOOTHING_FACTOR 0.2
```

```
uint8_t masterAddress[] = {0x88, 0x13, 0xBF, 0x03, 0x7D, 0xF0};

void setupSensor(MPU9250_WE &sensor, int index) {
  Serial.print("Initializing Sensor ");
  Serial.println(index + 1);
  if (!sensor.init()) {
    Serial.print("Sensor ");
    Serial.print(index + 1);
    Serial.println(" not responding!");
  } else {
    Serial.print("Sensor ");
    Serial.print(index + 1);
    Serial.println(" connected.");
  }

  delay(500);
  sensor.autoOffsets();
  sensor.enableGyrDLPF();
  sensor.setGyrDLPF(MPU9250_DLPF_6);
  sensor.setSampleRateDivider(5);
  sensor.setGyrRange(MPU9250_GYRO_RANGE_250);
  sensor.setAccRange(MPU9250_ACC_RANGE_2G);
  sensor.enableAccDLPF(true);
  sensor.setAccDLPF(MPU9250_DLPF_6);
  delay(200);
}

float calculatePostureAngle(const xyzFloat &g) {
  float norm = sqrt(g.x * g.x + g.y * g.y + g.z * g.z);
  float x = g.x / norm;
```

```
  float y = g.y / norm;
  float z = g.z / norm;
  return atan2(-x, sqrt(y * y + z * z)) * 180.0 / PI;
}

void sendAngles() {
  String payload = "";
  for (int i = 0; i < 6; i++) {
    xyzFloat g = sensors[i].getGValues();
    float rawAngle = calculatePostureAngle(g);
    smoothedAngles[i] = (SMOOTHING_FACTOR * rawAngle) + ((1 -
       SMOOTHING_FACTOR) * smoothedAngles[i]);
    payload += String(smoothedAngles[i], 2);
    if (i < 5) payload += ",";
  }

  esp_now_send(masterAddress, (uint8_t *)payload.c_str(), payload.
     length());
  Serial.print("Sent: ");
  Serial.println(payload);
}

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW init failed!");
    return;
  }
  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, masterAddress, 6);
```

```
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }

  for (int i = 0; i < 6; i++) {
    setupSensor(sensors[i], i);
  }

  Serial.println("Setup complete, sending data...");
}

void loop() {
  sendAngles();
  delay(1000);  // 1-second interval
}
```

Code for ESP32 Slave 2

```
#include <WiFi.h>
#include <esp_now.h>
#include <MPU9250_WE.h>
#include <SPI.h>
#include <math.h>

// Chip Select pins for 5 sensors
const int csPins[5] = {5, 17, 16, 4, 2};  // Update as per your
   wiring
MPU9250_WE sensors[5] = {
```

```
  MPU9250_WE(&SPI, csPins[0], true),
  MPU9250_WE(&SPI, csPins[1], true),
  MPU9250_WE(&SPI, csPins[2], true),
  MPU9250_WE(&SPI, csPins[3], true),
  MPU9250_WE(&SPI, csPins[4], true),
};

float smoothedAngles[5] = {0};
#define SMOOTHING_FACTOR 0.2

// MAC address of the Master ESP32
uint8_t masterAddress[] = {0x88, 0x13, 0xBF, 0x03, 0x7D, 0xF0};

void setupSensor(MPU9250_WE &sensor, int index) {
  Serial.print("Initializing Sensor ");
  Serial.println(index + 1);
  if (!sensor.init()) {
    Serial.print("Sensor ");
    Serial.print(index + 1);
    Serial.println(" not responding!");
  } else {
    Serial.print("Sensor ");
    Serial.print(index + 1);
    Serial.println(" connected.");
  }

  delay(500);
  sensor.autoOffsets();
  sensor.enableGyrDLPF();
  sensor.setGyrDLPF(MPU9250_DLPF_6);
  sensor.setSampleRateDivider(5);
```

```
  sensor.setGyrRange(MPU9250_GYRO_RANGE_250);
  sensor.setAccRange(MPU9250_ACC_RANGE_2G);
  sensor.enableAccDLPF(true);
  sensor.setAccDLPF(MPU9250_DLPF_6);
  delay(200);
}

float calculatePostureAngle(const xyzFloat &g) {
  float norm = sqrt(g.x * g.x + g.y * g.y + g.z * g.z);
  float x = g.x / norm;
  float y = g.y / norm;
  float z = g.z / norm;
  return atan2(-x, sqrt(y * y + z * z)) * 180.0 / PI;
}

void sendAngles() {
  String payload = "";
  for (int i = 0; i < 5; i++) {
    xyzFloat g = sensors[i].getGValues();
    float rawAngle = calculatePostureAngle(g);
    smoothedAngles[i] = (SMOOTHING_FACTOR * rawAngle) + ((1 -
       SMOOTHING_FACTOR) * smoothedAngles[i]);
    payload += String(smoothedAngles[i], 2);
    if (i < 4) payload += ",";
  }

  esp_now_send(masterAddress, (uint8_t *)payload.c_str(), payload.
     length());
  Serial.print("Sent: ");
  Serial.println(payload);
}
```

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW init failed!");
    return;
  }

  esp_now_peer_info_t peerInfo = {};
  memcpy(peerInfo.peer_addr, masterAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;

  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add peer");
    return;
  }

  for (int i = 0; i < 5; i++) {
    setupSensor(sensors[i], i);
  }

  Serial.println("Slave 2 setup complete. Sending data...");
}

void loop() {
  sendAngles();
  delay(1000);  // 1-second interval
}
```

ESP32 Master Code

```
#include <WiFi.h>
#include <esp_now.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

// WiFi Credentials
#define WIFI_SSID "Your_WiFi_SSID"
#define WIFI_PASS "Your_WiFi_Password"

// Adafruit IO Configuration
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "Your_Adafruit_Username"
#define AIO_KEY         "Your_Adafruit_Key"

// MAC Addresses of Slaves
uint8_t slave1Address[] = {0x08, 0xA6, 0xF7, 0xBC, 0x98, 0xA8};  //
   Upper Body (6 sensors)
uint8_t slave2Address[] = {0x2C, 0xBC, 0xBB, 0x0C, 0x63, 0x14};  //
   Lower Body (5 sensors)

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
   AIO_USERNAME, AIO_KEY);
Adafruit_MQTT_Publish postureFeed = Adafruit_MQTT_Publish(&mqtt,
   AIO_USERNAME "/feeds/body-posture");

void connectWiFi() {
  Serial.print("Connecting to ");
  Serial.println(WIFI_SSID);
```

```
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nWiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void connectMQTT() {
  int8_t ret;
  while ((ret = mqtt.connect()) != 0) {
    Serial.println(mqtt.connectErrorString(ret));
    Serial.println("Retrying MQTT connection in 5 seconds...");
    mqtt.disconnect();
    delay(5000);
  }
  Serial.println("MQTT Connected!");
}

void OnDataRecv(const esp_now_recv_info_t *recv_info, const uint8_t *
   data, int len) {
  const uint8_t *mac = recv_info->src_addr;
  char macStr[18];

  snprintf(macStr, sizeof(macStr), "%02X:%02X:%02X:%02X:%02X:%02X",
           mac[0], mac[1], mac[2], mac[3], mac[4], mac[5]);
```

```
  String receivedData = "";
  for (int i = 0; i < len; i++) {
    receivedData += (char)data[i];
  }

  // Create JSON payload
  String payload = "{";
  payload += "\"mac\":\"" + String(macStr) + "\",";
  payload += "\"data\":\"" + receivedData + "\"";
  payload += "}";

  // Publish to Adafruit IO
  if (!postureFeed.publish(payload.c_str())) {
    Serial.println("Failed to publish to Adafruit IO");
  } else {
    Serial.println("Data published to Adafruit IO");
  }

  // Serial output
  if (memcmp(mac, slave1Address, 6) == 0) {
    Serial.println("[Upper Body - Slave 1] Angles: " + receivedData);
  } else if (memcmp(mac, slave2Address, 6) == 0) {
    Serial.println("[Lower Body - Slave 2] Angles: " + receivedData);
  } else {
    Serial.println("Unknown sender (" + String(macStr) + ") Data: " +
        receivedData);
  }
}

void setup() {
  Serial.begin(115200);
```

```
  // Connect to WiFi
  connectWiFi();

  // Initialize ESP-NOW
  WiFi.mode(WIFI_STA);
  if (esp_now_init() != ESP_OK) {
    Serial.println("ESP-NOW initialization failed!");
    return;
  }

  // Register callback
  esp_now_register_recv_cb(OnDataRecv);

  // Connect to MQTT
  connectMQTT();

  Serial.println("Master ESP32 ready (ESP-NOW + Adafruit IO)");
}

void loop() {
  // Maintain MQTT connection
  if (!mqtt.ping()) {
    mqtt.disconnect();
    connectMQTT();
  }

  delay(1000); // Maintain connection without blocking
}
```

ESP32 MAC Address Reader Code

```
#include <WiFi.h>
#include <esp_wifi.h>

void readMacAddress(){
  uint8_t baseMac[6];
  esp_err_t ret = esp_wifi_get_mac(WIFI_IF_STA, baseMac);
  if (ret == ESP_OK) {
    Serial.printf("%02x:%02x:%02x:%02x:%02x:%02x\n",
                  baseMac[0], baseMac[1], baseMac[2],
                  baseMac[3], baseMac[4], baseMac[5]);
  } else {
    Serial.println("Failed to read MAC address");
  }
}

void setup(){
  Serial.begin(1150);

  WiFi.mode(WIFI_STA);
  WiFi.begin();

  Serial.print("[DEFAULT] ESP32 Board MAC Address: ");
  readMacAddress();
}

void loop(){
}
```

BHAVYA JAIN

+91-9463876695 || bhavyajain1210@gmail.com || **GitHub** || **LinkedIn**

EDUCATION

B. Tech	Bharati Vidyapeeth College of Engineering, Pune	2021-Present	89.5%
Higher Secondary Certificate Exam	Woodland Overseas School, Hoshiarpur	2021	85.8%
Senior Secondary Examination	Woodland Overseas School, Hoshiarpur	2019	91.2%

PROJECTS

Type Rush	• Engineered a scoring system to evaluate **typing speed** on a laptop using **HTML, CSS, and JavaScript**. Calculates **Words per Minute** (WPM), **Characters per Minute** (CPM), **mistakes**, and **accuracy**. Provides metrics with precision up to **2 decimal places** for enhanced real-time feedback. • Incorporated a real-time feedback feature that visually indicates **correct** and **incorrect** keystrokes by changing letter colors to **green** or **red**, improving user awareness and learning during gameplay. • **https://bhavyajain12.github.io/Type-Rush/**
Simon Game	• Programmed a memory-based game using **HTML, CSS**, and **JavaScript** that generates random patterns for players to memorize, increasing difficulty progressively and enhancing cognitive abilities. • Instituted a dynamic **scoring system** to track player performance and encourage consistent improvement; users must input the **correct sequence** to advance, while **incorrect sequences** require restarting from the beginning, ensuring a challenging and engaging experience. • Utilized **event listeners and callbacks** to manage user interactions and game logic, integrating **4 distinct sound effects** for blocks to enhance memory, engagement, and immersive gameplay. • **https://bhavyajain12.github.io/SimonGame/**
Automated Car Parking System	• Constructed an **Automated Car Parking System** with **Arduino Uno**, combining **2 IR sensors** for vehicle detection and **1 servo motor** for gate control, enhancing overall parking efficiency. • Devised a **16x2 LCD I2C** display for **real-time parking updates**, enhancing user experience. • Configured the system to **automatically** detect **available slots** and prevent entry when full, optimizing parking **space utilization**, **improving traffic flow**, and enhancing overall convenience. • **https://github.com/Bhavyajain12/Automated-Car-Parking-System**

SKILLS

Languages	C++ (Preferred), C, SQL	**Database**	MySQL
Developer tools	GitHub, Visual Studio, Postman, CodeBlocks, MATLAB, Arduino IDE		

Data Structures & Algorithm	Object Oriented Programming	Software Engineering

AREA OF INTEREST

Web Development	Problem Solving	Sports

EXPERIENCE

Tata Consultancy Services	• Successfully placed at Tata Consultancy Services (TCS) , aligned with the IoT and Digital Engineering domain, contributing to the development of intelligent, connected solutions.
Cognifyz Technologies	• Developed a React **weather app** that fetches **real-time data** from an API, allowing users to check weather for any location. It also displays additional details such as date, time, and wind conditions.

CERTIFICATIONS

C++	• Accredited in **Learn C++ Programming beginner to advance** in C++ from UDEMY.
Web Development	• Certified in **The Complete Web Development Bootcamp** from UDEMY.
Micro-Sensors	• Achieved certification in **A Brief Introduction to Micro-Sensors** from NPTEL.
JavaScript	• Completed **Fundamentals of JavaScript through rock paper-scissors** from Cognitive Class.

ACHIEVEMENTSs

Abacus	• Secured second prize in the Brain Boosters Annual Abacus Competition (Level 2) in Ambala Cantt.
Scouts and Guides	• Awarded a training certificate for **Scouts and Guides by Hindustan Scouts and Guides**, Punjab State, recognizing leadership and teamwork skills developed during the program.

Satyam Mishra

+91 9910614849 ◦ mishrasatyam0602@gmail.com
◦ LinkedIn ◦ GitHub

EDUCATION

B.TECH	Bharati Vidyapeeth University's COE , Pune Maharashtra [2022- 25]	**CGPA-9.07**
DIPLOMA	PUSA Institue Of Technology , New Delhi [2019-22]	**81.8%**
Xth	Kendriya Vidyalaya, AFS , Tughlakabad , New Delhi [2019]	**87%**

SKILLS

Electronics	Basic Electronics, Digital Electronics, Embedded Systems, Designing.
Lab Equipment	CRO, Multimeter, Signal Generator, Power Supply, Oscilloscope.
Software Tools	Arduino IDE, Altium Designer, Keil, Multisim, Proteus, VS Code, Xilinx.
Technical Skills	Components identification , Components Integration, Testing, Debugging, C Programming.

EXPERIENCE

Young Research Fellow (Hardware) –iHub-Data, IIIT Hyderabad — Present

- Designing multilayer PCB for high-performance and application-specific PCBs, ensuring efficient component placement and signal integrity.
- Research and development in embedded system in Healthcare Domain.

Research Intern– SSPL, DRDO , Delhi — June-Aug 2024

- Researched the growth of advanced heterostructures using Metal Organic Chemical Vapor Deposition (MOCVD) technology.
- Gained knowledge of MOCVD technology for the controlled deposition of thin films and heterostructures.

Internship – SSPL, DRDO , Delhi — June-July 2021

- Characterized high-electron-mobility-transistors (HEMT) using AlGaN/GaN heterostructures for advanced high-frequency and power electronics applications.

PROJECTS

Weather Monitoring IoT System:

- **Technologies:** Node-RED, ThingSpeak, DHT11 Sensor, ESP8266 Microcontroller.
- Designed an IoT-based system to monitor temperature and humidity with real-time cloud integration.
- Utilized ESP8266 for wireless data transmission and Node-RED for data visualization.

Automated Car Parking Management System:

- **Technologies:** Arduino Uno, LCD Display, Servo Motor, IR Sensors.
- Developed a smart parking system with IR sensors for spot detection and servo motors for automated barrier control.
- Displayed parking status on an LCD screen, enabling efficient vehicle guidance.

Wi-Fi Controlled Robot Car:

- **Technologies:** ESP8266, Motor Driver, Gear Motors, Mobile Application.
- Built a remotely controlled car with ESP8266 for Wi-Fi communication and navigation via a mobile app.
- Implemented precise motion control using motor drivers and Gear motors.

ELECTO-STEP: Mechanical Energy Harvesting System:

- **Technologies:** Piezoelectric Plates, Rectifier Circuit, Energy Storage Module.
- Designed a wearable system to convert walking energy into electricity using piezoelectric plates.
- Developed a detachable energy storage module for portable device charging.

RAM PRAKASH

Badarpur , New Delhi 110044

+91 8882059044 # ramprakash130802@gmail.com

Summary

Focused and thorough engineering student driven by a passion for continuous skill improvement and teamwork. With strong coordination abilities and a knack for prioritization, I adeptly manage various tasks concurrently. Seeking to apply my talents in the field of ECE and IT engineering, I aspire to build a fulfilling career in the industry.

Education

Bharati Vidyapeeth Deemed University, Pune – Expected in 06/2025 Bachelor of Technology, 4th year, ECE (8 cgpa till 6th semester)

Pusa Institute Of Techonology , New Delhi – in 06/2022 – 75%

Notre Dame School, Badarpur, New Delhi – 04/2018 – CBSE – 70%

Projects

1. Automated Parking System
2. Mobile controlled Wifi car
3. Simulation of Real Time Lane Detection using Python
4. Two Digit Counter Using Arduino & 7 Segment Display & Push Button
5. Power Bank charged by Walking

Vocational Traning

- **GaN/AlGaN/InAlGaN Research , DRDO , Delhi**
- **Training in DMRC , Delhi**

CORE Competencies

- **Observent**
- **Self-initiating**
- **Ambitious**
- **Contributive**
- **Quick to adapt**
- **Communication**
- **Team Work**