SMART POST STROKE UPPER LIMB REHABILITATION MONITORING AND FEEDBACK SYSTEM

FINAL YEAR PROJECT

Presented by

VISHNAKUMAR M

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANJALAI AMMAL MAHALINGAM ENGINEERING COLLEGE

THIRUVARUR, TAMIL NADU

DATE: 28 / 05 / 2025

My role in this project was to design the PCB and develop the real-time C software for hardware implementation, corresponding to the functioning of the ML model and loading into hardware.

Special Thanks to My Team Mates, ***YOGESH B, SIVAGANESH N, & VISHWA S***

Problem Statement

To enhance the effectiveness of rehabilitation for mild stroke patients undergoing home-based therapy and to resolve key feature deficiencies observed in existing rehabilitation solutions.

Introduction

Stroke is a highly debilitating chronic condition and a leading cause of death globally. It occurs when the blood supply to the brain is disrupted by a blockage or bleed, leading to rapid brain tissue damage from oxygen deprivation. This impairment affects vital bodily functions and can cause widespread complications across various body systems.

Stroke survivors often face mobility impairments, primarily affecting the upper extremities, which significantly impacts daily activities, leading to increased dependence and a reduced quality of life.

Stroke rehabilitation plays a crucial role in reducing motor disabilities and improving patients' quality of life. However, the severity of stroke-related impairments varies among individuals, necessitating personalized rehabilitation programs tailored to their specific neurological conditions.

A major challenge in stroke rehabilitation is the lack of proper assessment and feedback, especially for patients undergoing therapy at home, often resulting in insufficient and suboptimal rehabilitation outcomes.

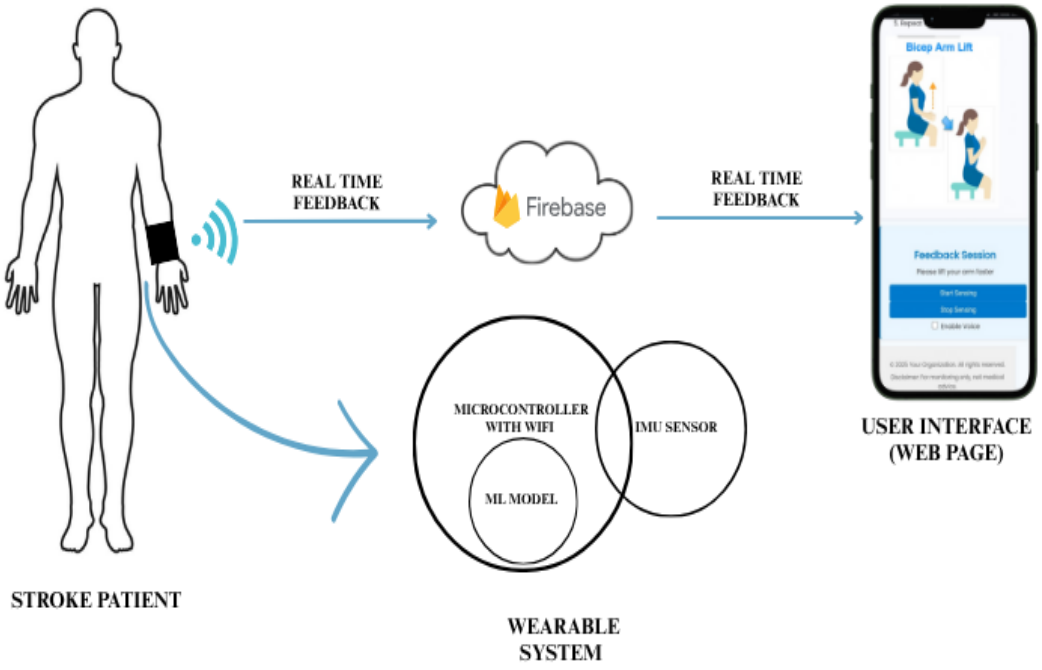
Aim

To improve the efficiency of rehabilitation for mild stroke patients by providing real-time feedback that helps them correct their exercises while performing self-directed therapy at home, thereby effectively reducing motor disabilities.

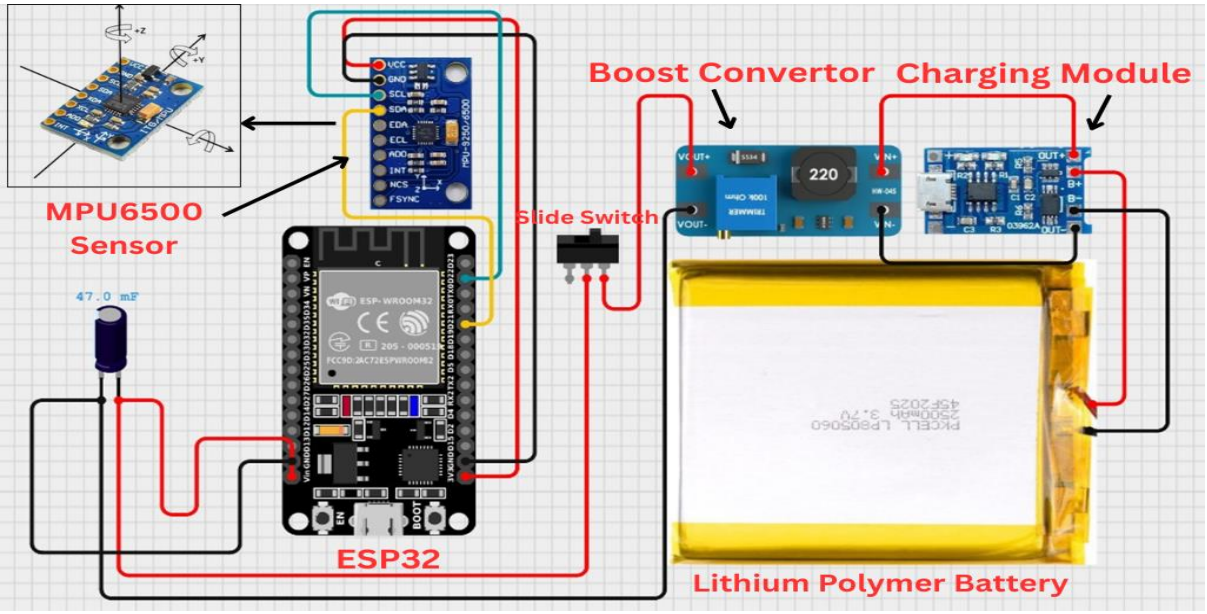
Key Focus Areas and Objectives

- **Focus on Mild Symptoms Patients:** Specifically target individuals who are partially affected and capable of performing self-directed rehabilitation.
- **Provide Real-time Feedback:** Utilize a Machine Learning (ML) model to provide feedback in both text and voice formats, significantly increasing the effectiveness of rehabilitation sessions.
- **Eliminate Location Constraints:** Incorporate Wi-Fi communication and a portable, wearable setup suitable for convenient home use.
- **Develop a User Interface:** Create an intuitive User Interface (UI) for patients to easily interact with and utilize the system.
- **Ensure Cost-Effectiveness and Broad Accessibility.**

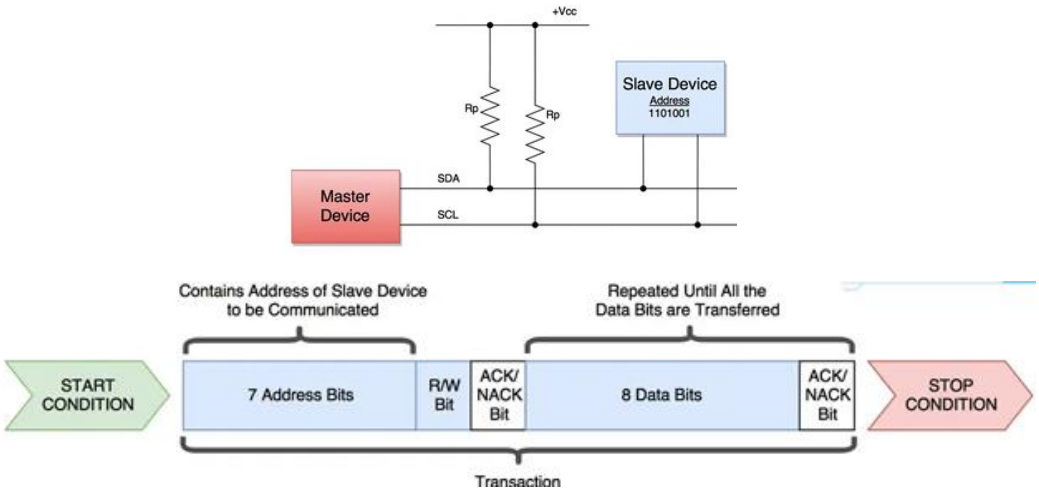
Proposed system Overview



Hardware Design



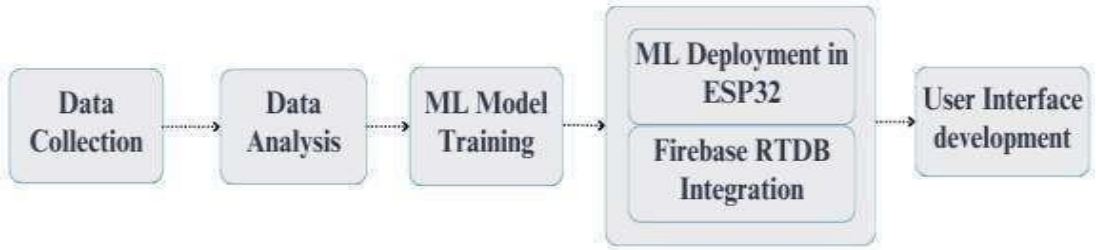
We have use I2C communication protocol between an ESP32 (master) and an MPU-6500 (slave). It have just two wires(SDA (data) and SCL (clock)) for communication



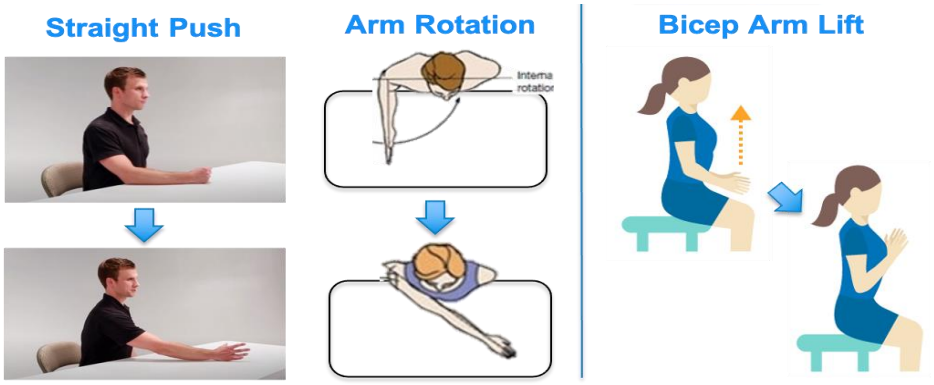
The process begins with the ESP32 sending a Start Condition, followed by the MPU-6500's 7-bit address with Read/Write bit. The MPU-6500 responds with an Acknowledgement (ACK). Next, the ESP32 sends the 8-bit address of the specific internal register it wants to access. Data is then transferred in 8-bit with each byte acknowledged. For 16-bit sensor data, like Accelerometer, gyroscope, and temperature readings are 16-bit values from the MPU-6500's ADCs. These are transferred as two separate 8-bit bytes (high and low bytes) over I2C, which the ESP32 then reassembles into a 16-bit value. The communication concludes with a Stop Condition from the ESP32, releasing the bus.

Hardware implementation:

Methodology:



Data collection:



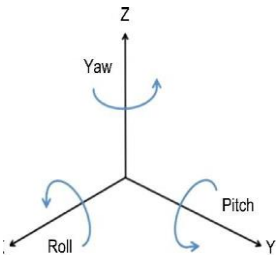
- We had selected three basic exercises for rehab, they are Straight Push, Arm Rotation, and Bicep Lifting
- To create a robust dataset, each exercise was performed both correctly and with common rehabilitation mistakes for feedbacks.
- The ESP32 was configured with specific digital low-pass filters for noise reduction, and automatic offset calibration for accuracy.
- We also apply complementary filter for calculating real-time pitch and roll angles by blending gyroscope and accelerometer readings.
- **It** merges accelerometer and gyroscope data to provide accurate and stable orientation. It favors the **gyroscope for short-term, rapid changes** (high-pass filtered) and the **accelerometer for long-term, stable tilt** (low-pass filtered). This method effectively counteracts gyroscope drift and accelerometer noise, delivering precise orientation tracking.

Pitch:

$$\text{pitch}_t = \alpha \cdot (\text{pitch}_{t-1} + g_x \cdot \Delta t) + (1-\alpha) \cdot \arctan 2(a_y, \sqrt{a_x^2 + a_z^2}) \cdot \frac{180}{\pi}$$

Roll:

$$\text{roll}_t = \alpha \cdot (\text{roll}_{t-1} + g_y \cdot \Delta t) + (1-\alpha) \cdot \arctan 2(-a_x, \sqrt{a_y^2 + a_z^2}) \cdot \frac{180}{\pi}$$



- Each motion instance was labeled in real-time for accurate annotation.
- The resulting motion data, including linear accelerations (Ax, Ay, Az), angular velocities (Gx, Gy, Gz), calculated orientation angles (Pitch, Roll), and distinct exercise labels, was serially transmitted as comma-separated values at approximately 100 Hz.

	A	B	C	D	E	F	G	H	I
1	Ax	Ay	Az	Gx	Gy	Gz	Pitch	Roll	Label
2	0.02	-0.12	1.07	7.58	-5.12	-6.19	0.63	-0.68	6
3	0.02	-0.11	1.09	7.85	-3.07	-6.16	0.32	-0.69	6
4	0.03	-0.13	1.09	7.97	-1.63	-5.54	-0.03	-0.73	6
5	0.02	-0.15	1.08	7.92	-1.35	-4.38	-0.41	-0.74	6
6	0.01	-0.16	1.07	7.52	-2.1	-3.74	-0.8	-0.72	6
7	0.01	-0.15	1.07	6.56	-4.42	-2.93	-1.16	-0.72	6
8	0.01	-0.15	1.07	5.38	-4.12	-3.11	-1.49	-0.71	6
9	0.02	-0.17	1.08	4.36	-4.77	-3.76	-1.86	-0.72	6
10	0.02	-0.17	1.09	4.24	-5.01	-4.02	-2.21	-0.75	6
11	0.02	-0.15	1.1	4.04	-3.37	-2.85	-2.48	-0.78	6
12	0.03	-0.13	1.1	4.92	-2.54	-2.54	-2.71	-0.82	6
13	0.03	-0.13	1.1	5.4	-1.22	-1.34	-2.9	-0.86	6

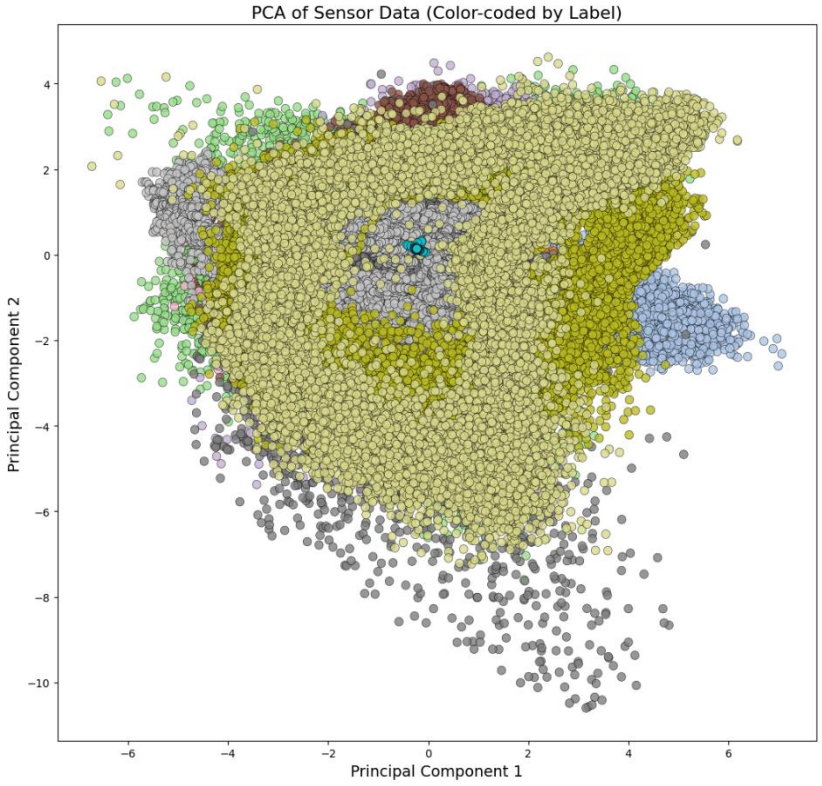
Data analysis:

Basically raw IMU sensor data is complex, because it is high dimensional data, so it leading to

- Increased computational cost,
- Risk of model overfitting.
- So we have employed **Principal Component Analysis (PCA)** for effective feature extraction and dimensionality reduction.
- PCA is a **unsupervised linear dimensionality reduction** technique.
- It transforms complex data into a **lower-dimensional space by** identifies **"Principal Components"**– new, uncorrelated variables that capture the directions of maximum variance in the data.
- **Key Benefit:** Retains the most significant information while reducing the number of features.

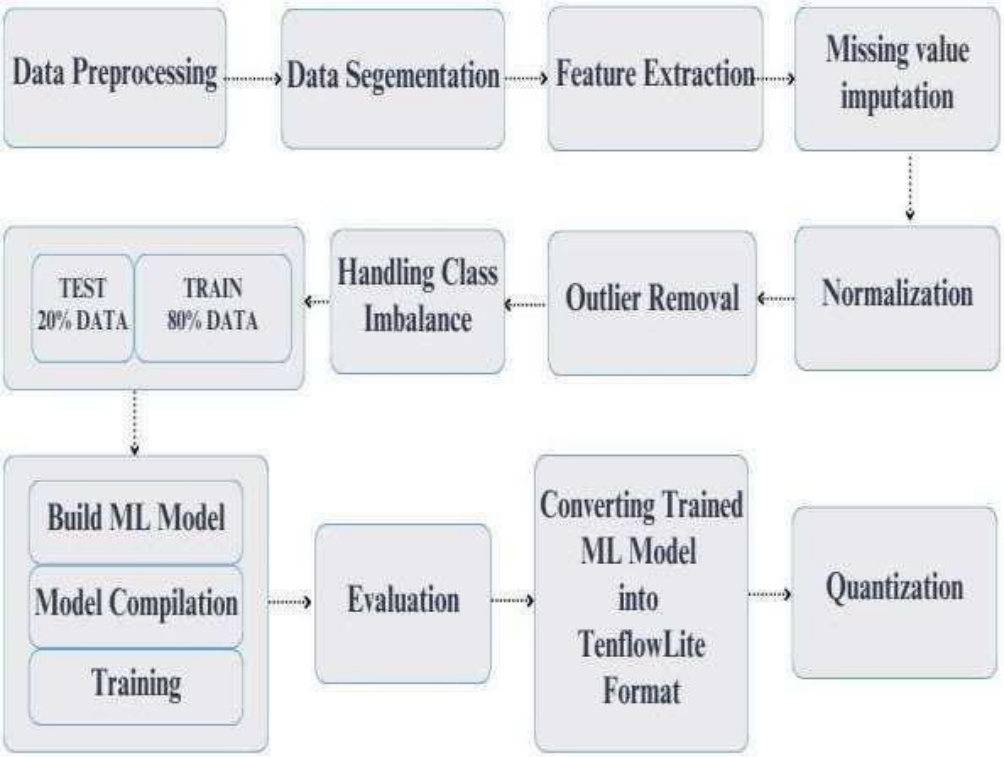
How PCA Works:

1. **Standardize Data:** Ensure all features contribute equally (mean 0, std dev 1).
2. **Find Relationships:** Compute the **Covariance Matrix** to understand how features vary together.
3. **Identify Orthogonal Axes:** Calculate **Eigenvectors** (our Principal Components) and **Eigenvalues** (representing variance captured) from the covariance matrix.
4. **Select & Transform:** Choose the top PCs (those with highest eigenvalues) to form a new, reduced dataset.



The results of the PCA were then visualized using label-specific scatter plots, illustrating the distribution of each class in the reduced dimensional space and color-coded by the original labels, providing a comprehensive overview of class separability in the PC1-PC2 plane. These visualizations offer insights into the inherent structure and potential for distinguishing between different exercise movements.

ML model training and deployment:



Data Preprocessing:

It transform the raw sensor data into suitable format for ml by loading multiple CSV files into a Pandas Data Frame and cleaning the data by removing missing values and perform label encoding, it convert the categorical exercise labels into numerical representations for neural network input.

Data Segmentation:

Applied **sliding window function** to segment time-series data into fixed-size windows, capturing temporal patterns.

Feature Extraction:

Calculated **13 statistical features** (e.g., Mean, Std Dev, Max, Min, Skewness, Kurtosis, RMS, Zero Crossings, Energy, IQR) for each of 8 sensor channels (accelerometer x, y, z; gyroscope x, y, z; pitch; roll).

Missing Value Imputation: Filled remaining missing values using median.

Normalization: Scaled features using Standard Scaler (zero mean, unit variance) for faster convergence.

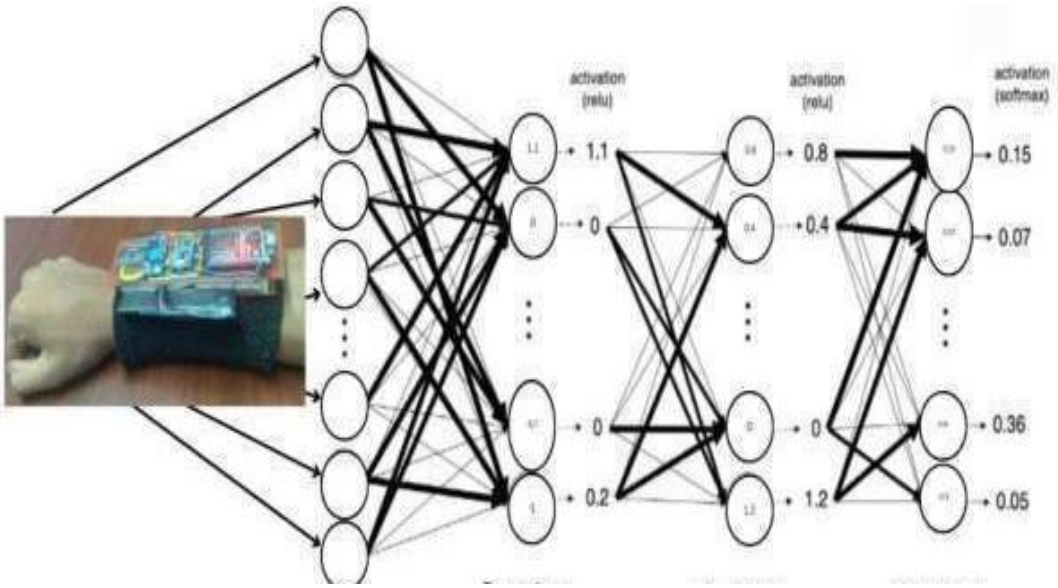
Outlier Removal: Applied **Local Outlier Factor (LOF)** to identify and remove anomalies.

Class Imbalance Handling: Used **SMOTE** to oversample minority classes, ensuring a balanced training set.

Train-Test Split: Dataset partitioned into 80% for training and 20% for testing.

ML Model and training:

The primary machine learning algorithm employed is a **Deep Neural Network (DNN)**, specifically a **Multi-Layer Perceptron (MLP)**, built using the Keras API with TensorFlow as the backend.



MLPs are artificial neural networks widely used for classification and regression tasks. An MLP consists of an input layer, one or more hidden layers, and an output layer, featuring fully connected (Dense) layers that transform input data.

Our model utilizes **ReLU activation** for the hidden layers to introduce non-linearity and **Softmax activation** for the output layer to convert predictions into a probability distribution.

Dropout regularization was incorporated to prevent overfitting. The model was trained using the **Adam optimizer** and **sparse categorical cross-entropy loss**

ML Model Deployment with TensorFlow Lite (TFLite)

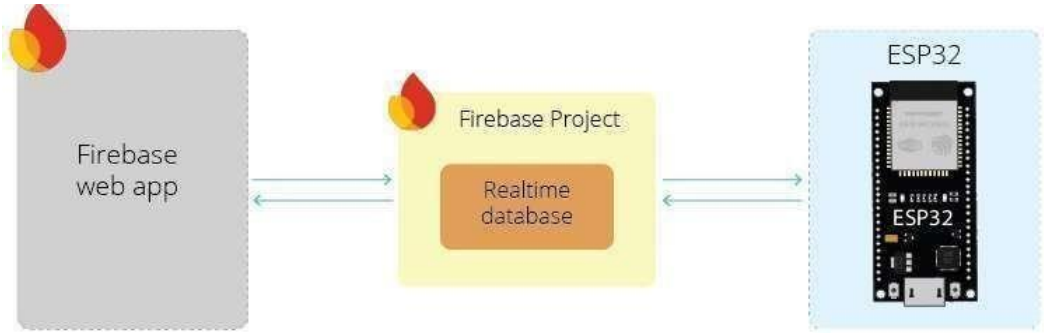
Conversion: The trained MLP model was converted into a **TensorFlow Lite (.tflite)** format and it enables **on-device machine learning inference** directly on resource-constrained devices (like ESP32) without cloud connectivity.

Quantization: Applied to reduce model size and improve inference speed by converting weights and activations from floating-point (F32) to lower-precision integers (INT8), maintaining accuracy.

FIREBASE RTDB INTEGRATION

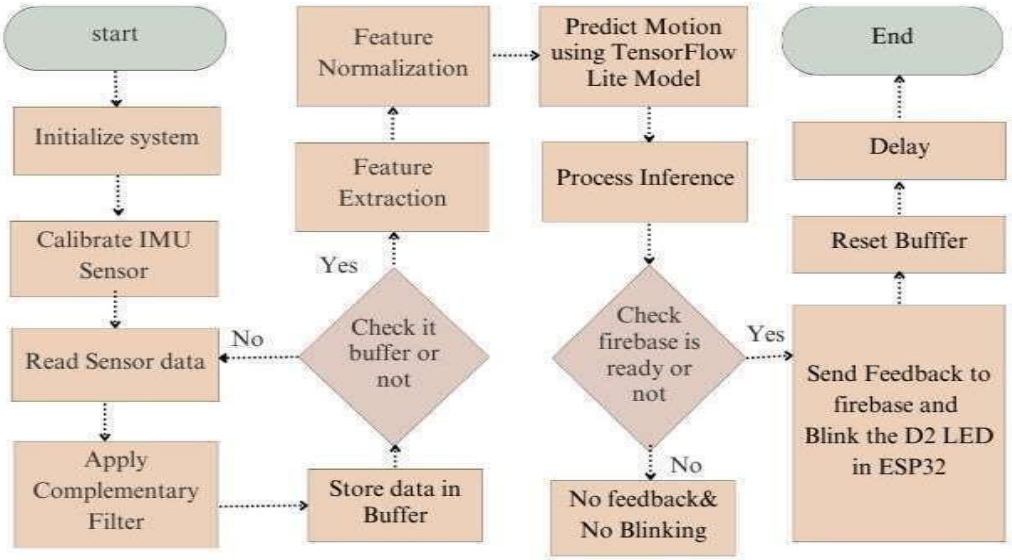
We have integrate the firebase's real time database for facilitates **real-time data transmission** from the ESP32 microcontroller to a user interface.

This enables **immediate feedback** to the patient during exercises and allows for **continuous monitoring of patient,** thereby improving the efficiency and effectiveness of home-based stroke rehabilitation.

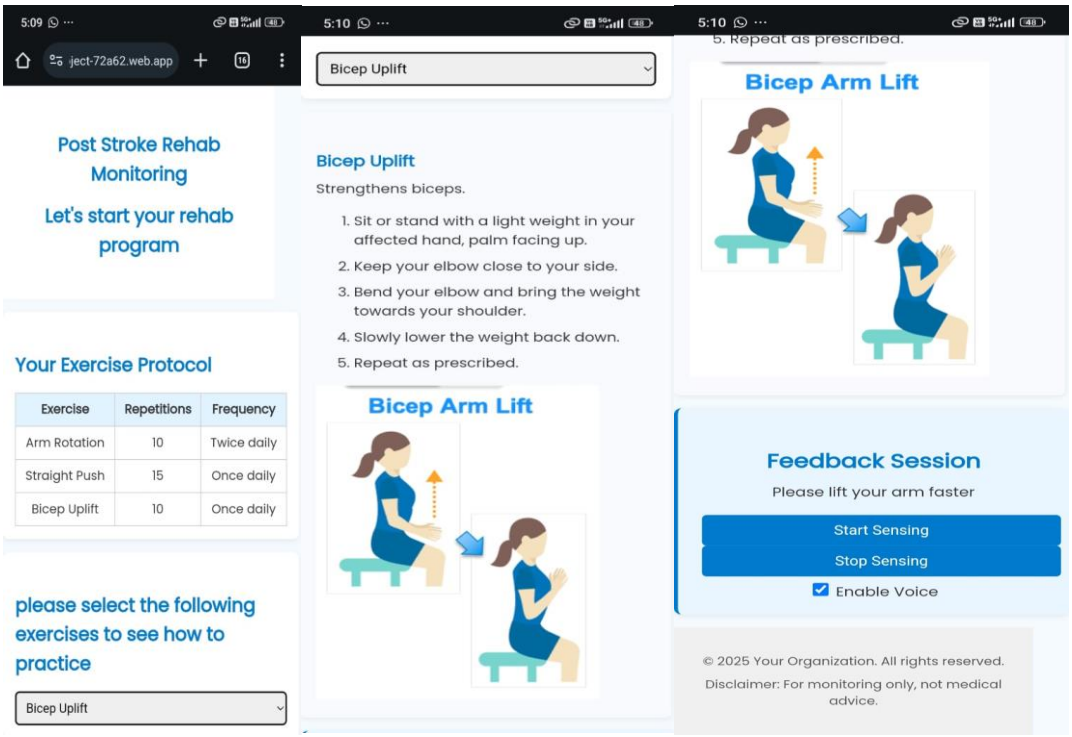


- **Realtime Database (RTDB) it has NoSQL, so it is cloud-hosted database** it offering **Real-time Data Synchronization**
- Flexible JSON-based structure and horizontal scalability.

WORKING FLOW



USER INTERFACE



This webpage is the user interface for our system.

It was by developed using HTML, CSS and Javascript.

It guides stroke patient through feedback by shown on in webpage and also give feedback through voice by enabling it.

It also have protocols and how to practice rehab.

start monitoring via the "Start" button.

The "Feedback" section displays dynamic instructions, with optional "Voice Feedback."

The "Stop" button allows users to end sessions.

This interactive platform aims to empower home-based rehabilitation by offering real-time visual and auditory guidance, potentially bridging the gap between clinic therapy and independent practice for stroke survivors.

CONCLUSION:

This project developed a post-stroke upper limb rehabilitation monitoring system. It utilizes an IMU to capture motion data, processes it to extract features, and employs a TensorFlow Lite model on a microcontroller for real-time motion recognition and user feedback. Exercise data is logged on Firebase for progress tracking and remote monitoring.

Key components include the MPU6500 for motion capture (accelerometer, gyroscope, pitch, roll), statistical feature extraction, a microcontroller-optimized TensorFlow Lite model for real-time classification, immediate user feedback for correct exercise performance, and Firebase integration for data logging and remote access.

FUTURE WORK:

1) Collect more diverse samples from different peoples to improve ML Model robustness.
2) Adding extra features for GUI functionalities
3) Validating the system with clinical experts and patients.
4) Improve the accuracy of the ML Model and use more stable sensor.
5) Improve the overall accuracy of the system by using more stable sensor and more powerful microcontroller.
6) We can able only achieved accuracy above 85% percentage, in future by using more power full components and precious software, we could achieve more accuracy results

---

TAKEAWAYS FROM THIS PROJECT

- Gained **hands-on soldering** experience and designed a basic, functional **PCB** layout.
- Applied the core principle of using **capacitors for power smoothing** and circuit stabilization.
- Gained knowledge of the I2C communication protocol.
- Gained knowledge of the **ESP32** and integration of a **motion tracking sensor**.
- Learned to integrate a **boost converter** and **charging module** with a **Li-Po battery**.
- Understood the critical role of filters (**Complementary Filter, Feature extraction, data analysis, segmentation, normalization, quantization**) and buffers in signal conditioning.
- Gained a full process idea for writing and implementing a **functional real-time C program** for hardware control with help of the **google gemini** and **reference project.**
