EMG-Controlled Adaptive Assistance for Wrist Rehabilitation: A Real-Time Control Platform

Dawood Aamar Mughal

October 16, 2025

Project Instructor: Prof. Dr. Claudio Castellini
Project Supervisor: Prof. Dr. rer. nat. Sabine Thürauf
Institution: FAU (Friedrich-Alexander-Universität)
Author's Email: dawood.a.mughal@fau.de

Abstract

This project presents the development of a real-time adaptive control platform for wrist rehabilitation that uses surface electromyography (sEMG) signals to provide responsive motor assistance. The system processes EMG data from forearm muscles to classify intended movements and translates these into smooth, controlled actions through an impedance-based control algorithm. Featuring a comprehensive safety system with software-enforced limits and a therapist-adjustable interface, the platform demonstrates the feasibility of EMG-driven assistance for rehabilitation applications. While not yet clinically validated, this work establishes a foundation for future adaptive rehabilitation devices by addressing key technical challenges in real-time signal processing, safety critical control, and human-robot interaction. The system represents a significant step toward personalized rehabilitation technology that can adapt to individual patient capabilities and progression. **Code Availability:** `https://gitlab.aibe.fau.de/airob/theses/research-lab-dawood-mughal`

1 Introduction

1.1 Background and Motivation

Wrist rehabilitation is a critical component of recovery for patients with neurological conditions, stroke, or musculoskeletal injuries. Traditional rehabilitation methods often rely on therapist guided exercises that can be labor intensive and lack quantitative feedback

[1]. This project addresses the growing need for automated, adaptive rehabilitation systems that can provide personalized assistance based on the patient's own muscle signals [2].

The integration of surface electromyography (sEMG) with robotic Exosuits represents a promising approach for creating responsive rehabilitation systems [3]. By interpreting the electrical activity of forearm muscles, the system can detect the user's movement intentions and provide appropriate assistance through the Exosuit. This approach enables more natural and engaging therapy sessions while reducing the burden on healthcare professionals.

1.2 Technical Challenges and Solutions

Developing an effective EMG controlled rehabilitation system involves addressing several complex technical challenges:

- **Real time Signal Processing:** EMG signals are inherently noisy and require sophisticated processing to extract meaningful information. The system must process these signals with low latency to provide natural feeling assistance. Our solution employs efficient feature extraction algorithms that can run in real time on mobile hardware.
- **Individual Variability:** EMG signal characteristics vary significantly between users due to factors like muscle anatomy, subcutaneous fat distribution, and electrode placement. Rather than using a one-size-fits-all approach, our system implements personalized model training that adapts to each user's unique signal patterns.
- **Safe Exosuit Control:** Providing physical assistance through an Exosuit requires rigorous safety measures. Our system implements multiple layers of protection, including parameter validation, position limits, and emergency stop mechanisms through the Android application to ensure user safety at all times.
- **System Integration:** Coordinating between mobile devices, training servers, and embedded controllers requires robust communication protocols. We developed a custom UDP based communication system that ensures reliable data exchange between all system components.

1.3 System Overview

The system employs a distributed architecture that separates concerns while maintaining tight integration:

- **Android Application:** Serves as the primary user interface, handling EMG data acquisition, real-time processing, and system control. The mobile platform provides accessibility and eliminates the need for specialized computing hardware.
- **Training Server:** Implements automated machine learning model optimization, exploring different feature combinations and window sizes to find the optimal configuration for each user. This server-based approach allows for computationally intensive training without burdening the mobile device.

- **Exosuit Controller:** Runs on a Raspberry Pi and handles real-time motor control with comprehensive safety monitoring. The controller implements position based assistance with multiple safety layers to prevent harmful movements.

This three-tier architecture enables each component to focus on its specialized task while maintaining seamless integration through well-defined communication protocols.

2 System Architecture

2.1 Hardware Components

The system integrates commercial off the shelf components with custom hardware to create a practical rehabilitation platform:

- **Myo Armband:** Selected for its 8-channel EMG sensing capability and built-in signal conditioning [8]. The armband provides 200Hz sampling with hardware bandpass filtering (5-200Hz), which effectively removes motion artifacts and high frequency noise. The wireless design and easy wearing and removal make it convenient and practical for clinical use.

- **Android Device:** Chosen as the primary interface due to its widespread availability and computational capabilities. Modern smartphones provide sufficient processing power for real-time EMG analysis while offering familiar touch interfaces for both therapists and patients.

- **Raspberry Pi 5:** Serves as the Exosuit control unit, providing adequate computational resources for real-time control algorithms while maintaining a small form factor. The Raspberry Pi's GPIO capabilities enable direct motor control through appropriate drivers.

- **AK60-6 Motors:** CubeMars motors were selected for their high torque-to-weight ratio (3Nm continuous torque) and precise position control capabilities. These motors provide sufficient force for wrist assistance while maintaining compact dimensions suitable for wearable applications.

- **Cable-driven Exosuit:** The custom mechanical design uses a cable driven transmission system that allows motor placement away from the hand, reducing the weight and inertia on the affected limb. This design choice improves comfort and allows for more natural movement.

2.2 Software Architecture

The software system employs a modular design that separates data acquisition, model training, and real-time control into distinct components. This separation of concerns enables independent development and testing of each subsystem while maintaining overall system coherence.

2.2.1 Android Application Architecture

The Android application follows the Model-View-ViewModel (MVVM) pattern, which provides clear separation between the user interface, business logic, and data management:

- **Data Acquisition Layer:** Handles Bluetooth communication with the Myo armband, receiving 8-channel EMG data at 200Hz. This layer implements buffering and preliminary signal validation to ensure data quality.
- **Processing Layer:** Implements real-time feature extraction and movement classification. This layer maintains a sliding window of EMG data and computes features only when sufficient data is available, balancing latency and classification accuracy.
- **Communication Layer:** Manages UDP communication with both the training server and Exosuit controller. This layer implements reliability mechanisms including sequence numbering, acknowledgment, and retransmission for critical messages.
- **User Interface Layer:** Provides intuitive controls for system operation, training guidance, and real-time feedback. The interface is designed to be accessible for both therapists and patients with varying technical proficiency.

2.2.2 Training Server Design

The Python-based training server implements a comprehensive model optimization pipeline:

- **Data Management:** Handles chunked data transmission from the Android application, reassembling complete datasets from multiple UDP packets. This approach allows training with large EMG datasets that exceed typical UDP packet size limits.
- **Parameter Exploration:** Systematically tests different combinations of window sizes and feature sets to identify optimal configurations. This exhaustive search ensures that the system can adapt to individual user characteristics.
- **Model Training:** Supports both Ridge regression and neural network approaches, allowing performance comparison and selection of the most appropriate model type for each user.
- **Model Distribution:** Packages trained models and their associated parameters for transmission back to the Android application.

2.2.3 Exosuit Controller Architecture

The Raspberry Pi controller implements safety-critical real-time control:

- **Communication Interface:** Listens for control commands and parameter updates from the Android application, implementing validation and acknowledgment mechanisms.
- **Safety Supervisor:** Continuously monitors the system state and intervenes when necessary to prevent unsafe conditions, including enforcing position and velocity limits.

- **Motor Control:** Implements position-based impedance control using the pyCandleMAB library, providing smooth and natural movement assistance.
- **Parameter Validation:** Validates all incoming parameters against safety bounds defined in the parameter registry, ensuring that only safe commands are executed.

3 Methods

3.1 EMG Signal Processing

3.1.1 Signal Acquisition and Preprocessing

The system acquires EMG signals through the Myo armband, which provides several advantages for rehabilitation applications. The built-in hardware filtering (20-500Hz bandpass) effectively removes both low-frequency motion artifacts and high-frequency noise, providing a clean signal for processing. The Android application utilizes the Myonnaise library [4] for Bluetooth communication and data acquisition from the Myo armband, providing reliable 200Hz sampling of 8-channel EMG data. The 200Hz sampling rate represents a practical compromise between temporal resolution and computational requirements, capturing the relevant EMG frequency content while maintaining real-time performance on mobile hardware.

A critical design decision was to rely on the Myo's hardware filtering without implementing additional software filtering stages. This approach reduces computational load and latency, which are crucial for real-time operation. Our analysis determined that the combination of hardware filtering and robust feature extraction provides sufficient noise immunity for classification purposes. Specifically, utilizing only the **Root Mean Square (RMS) feature with a 60-sample window size** yielded highly satisfactory classification results, allowing the simplified processing pipeline to maintain the low latency needed for natural-feeling assistance.

3.1.2 Feature Extraction Strategy

The feature extraction module transforms raw EMG signals into meaningful representations for movement classification. Rather than using a fixed feature set, our system explores multiple feature combinations to find the optimal configuration for each user. This personalized approach recognizes that different users may exhibit distinct EMG patterns that are best captured by different feature sets [5].

The system computes six time domain features that capture different aspects of the EMG signal [6]:

- **Root Mean Square (RMS)** quantifies signal power and correlates well with muscle force production. This feature is particularly effective for detecting muscle activation levels and is widely used in EMG analysis due to its robustness to outliers.
- **Mean Absolute Value (MAV)** provides a computationally efficient measure of signal amplitude. Although similar to RMS, MAV is less computationally intensive and provides stable performance across different signal conditions.

- **Variance (VAR)** measures the variability of the signal around its mean, capturing the spread of the distribution of EMG signals. This feature helps distinguish between different force levels and muscle activation patterns.
- **Waveform Length (WL)** accumulates absolute differences between consecutive samples, effectively measuring signal complexity and frequency content. WL increases with both signal amplitude and frequency content, providing complementary information to amplitude-based features.
- **Zero Crossings (ZC)** counts signal sign changes, providing information about frequency characteristics. A threshold is applied to prevent counting noise induced crossings, making this feature robust to low-amplitude fluctuations.
- **Slope Sign Changes (SSC)** detect changes in the direction of the signal slope, complementing zero crossings by capturing different aspects of the signal morphology. SSC is particularly sensitive to complex signal patterns that occur during coordinated muscle activations.

The mathematical formulations for these features ensure they can be computed efficiently in real-time:

- **Root Mean Square (RMS):**

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

- **Mean Absolute Value (MAV):**

$$MAV = \frac{1}{N}\sum_{i=1}^{N} |x_i|$$

- **Variance (VAR):**

$$VAR = \frac{1}{N}\sum_{i=1}^{N} (x_i - \bar{x})^2$$

- **Waveform Length (WL):**

$$WL = \sum_{i=1}^{N-1} |x_{i+1} - x_i|$$

- **Zero Crossings (ZC):**

$$ZC = \sum_{i=1}^{N-1} \mathbf{1}\big(x_i \cdot x_{i+1} < 0 \ \wedge \ |x_i - x_{i+1}| \geq \theta\big)$$

  where θ is a small threshold (e.g., 0.01) to avoid noise-induced crossings.

- **Slope Sign Changes (SSC):**

$$SSC = \sum_{i=2}^{N-1} \mathbf{1}\big((x_i - x_{i-1})(x_i - x_{i+1}) > 0 \ \wedge \ (|x_i - x_{i-1}| \geq \theta \ \vee \ |x_i - x_{i+1}| \geq \theta)\big)$$

Where x_i represents the EMG signal samples, N is the window size, and θ is a noise threshold (typically 0.01-0.05) to prevent counting small fluctuations caused by noise.

3.1.3 Window Size Selection

The system explores window sizes from 10 to 60 samples (50-300ms at 200Hz) to balance temporal resolution and feature stability. Smaller windows provide better temporal resolution but may suffer from increased variance, while larger windows offer more stable features at the cost of increased latency. The optimal window size depends on the characteristics of the movements being classified and the individual user's EMG patterns.

3.2 Machine Learning Framework

3.2.1 Multi-Output Ridge Regression

The system employs a multi-output Ridge regression approach for movement classification, which provides several advantages for rehabilitation applications [7]. Unlike traditional classification methods that output a single class label, this approach generates probability distributions across all four movement classes (isometric, extension, flexion, rest). This probabilistic output enables smooth transitions between movements and provides more nuanced control of the Exosuit.

The Ridge regression model is particularly suitable for this application due to its regularization properties, which prevent overfitting a critical consideration given the high-dimensional feature space relative to the typically limited training data available in rehabilitation settings. The L2 regularization (controlled by the alpha parameter) adds a penalty term to the loss function that shrinks coefficient estimates toward zero, reducing model variance at the cost of a slight increase in bias. This trade-off is beneficial for EMG classification, where we prioritize robust performance over perfect training fit.

The training process involves creating four separate Ridge regression models, one for each movement class. Each model learns to predict the probability of its respective class based on the extracted EMG features. This independent training approach allows each model to focus on patterns specific to its class, rather than trying to capture all class differences in a single complex model.

During real-time operation, all four models process the same feature vector simultaneously, generating independent probability estimates. These estimates are then normalized using a softmax function to ensure they form a valid probability distribution (summing to 1.0). This approach provides natural handling of uncertainty and enables smooth transitions between movement states.

3.2.2 Systematic Parameter Exploration

A key innovation in our system is the systematic exploration of different parameter combinations during training. The training server evaluates multiple configurations (feature sets $\times$ window sizes) to identify the optimal setup for each user, ensuring the model discovers the best-performing parameters for personalized control.

The feature set progression follows a logical complexity gradient:

1. **Basic Features:** Individual features (RMS or MAV alone) for simple, computationally efficient processing

2. **Standard Combination:** RMS combined with MAV, providing both power and amplitude information

3. **Enhanced Sets:** Progressive addition of variance, waveform length, zero crossings, and slope sign changes

4. **Complete Feature Set:** All six features combined for maximum information capture

This systematic approach ensures that the system can adapt to the substantial inter-subject variability in EMG signal characteristics, providing personalized performance without requiring manual parameter tuning.

3.2.3 Alternative MLP Approach

As a comparative approach, the system also supports Multi-Layer Perceptron (MLP) neural networks converted to TensorFlow Lite format. The MLP approach offers greater model capacity and can capture more complex, non-linear relationships in the EMG data. However, this comes at the cost of increased computational requirements and reduced interpretability compared to Ridge regression.

The MLP implementation uses a simple architecture with two hidden layers (64 and 32 neurons) and softmax output activation for four-class classification. While more computationally intensive, the MLP approach provides a valuable alternative for users whose EMG patterns may not be well-captured by linear models.

3.3 Safety System Design

3.3.1 Parameter Validation Framework

The system implements a comprehensive parameter validation framework that ensures all control parameters remain within safe operating bounds. This is implemented through a parameter registry that defines acceptable ranges for each configurable parameter :

- **Position Control Parameters:** Stiffness (Kp) and damping (Kd) gains are constrained to prevent overly aggressive control that could cause discomfort or injury.

- **Movement Limits:** Position and velocity limits are derived from biomechanical studies of normal wrist range of motion and movement speeds.

- **Assistance Parameters:** Strength scaling factors and movement thresholds are bounded to ensure appropriate assistance levels.

When the Android application sends parameter updates to the Exosuit controller, each parameter is validated against the registry. Values outside acceptable ranges are clamped to the nearest safe value rather than rejected outright. This approach maintains system operation while ensuring safety, which is particularly important in rehabilitation contexts where continuous therapy is beneficial.

3.3.2 Real-time Safety Monitoring

During operation, the system continuously monitors multiple safety metrics:

- **Position Limit Enforcement:** Every commanded position is checked against the configured limits before being sent to the motors. This prevents movements beyond the safe anatomical range of the wrist.

- **Velocity Limiting:** The system computes the required velocity to reach commanded positions and ensures this velocity does not exceed safe limits. This prevents sudden, jerky movements that could cause discomfort or injury.

These safety mechanisms work together to ensure that the system provides helpful assistance while preventing any potentially harmful movements.

3.4 Real-time Control Algorithm

3.4.1 Position-Based Assistance Logic

The core control algorithm implements position-based assistance that translates EMG classification results into smooth motor movements. The algorithm operates in a continuous loop, ensuring responsive assistance while maintaining safety constraints.

The control logic follows these key steps:

1. **Movement State Detection:** The system continuously monitors the classified movement state (flexion, extension, isometric, or rest) and the associated confidence strength (0.0 to 1.0).

2. **Threshold Filtering:** A minimum movement threshold filters out weak EMG activations, preventing unintended movements and improving system stability:

$$\text{effective\_strength} = \begin{cases} \text{raw\_strength} & \text{if raw\_strength} \geq \theta_{\min} \\ 0.0 & \text{otherwise} \end{cases}$$

   where $\theta_{\min}$ is the configurable minimum movement threshold.

3. **Target Position Calculation:** Based on the detected movement, target positions are computed using scaled position limits:

   - **Flexion:** $\text{target} = L_{\text{upper}} \times s_{\text{effective}} \times \alpha_{\text{flex}} \times v_{\text{movement}}$
   - **Extension:** $\text{target} = L_{\text{lower}} \times s_{\text{effective}} \times \alpha_{\text{ext}} \times v_{\text{movement}}$
   - **Isometric/Rest:** $\text{target} = 0.0$

   where L represents position limits, s the effective strength, α strength scaling factors, and v movement speed.

4. **Deadzone Application:** A configurable deadzone around neutral position prevents motor jitter during rest states:

$$\text{target} = \begin{cases} 0.0 & \text{if } |\text{target}| < \theta_{\text{deadzone}} \\ \text{target} & \text{otherwise} \end{cases}$$

5. **Position Smoothing:** Exponential smoothing ensures gradual transitions between positions:

$$p_{\text{smoothed}} = (1 - \beta) \times p_{\text{previous}} + \beta \times p_{\text{target}}$$

   where β is the smoothing factor (0.05 default).

6. **Safety Enforcement:** Final positions are clamped within biomechanically safe limits before motor command transmission.

3.4.2 Asynchronous Implementation

The control algorithm is implemented as an asynchronous coroutine to handle multiple concurrent tasks:

- **Non-blocking Operation:** The loop runs independently while allowing other system tasks (communication, safety monitoring) to execute concurrently.
- **Connection State Management:** The controller checks motor connection status each iteration, gracefully handling disconnections without system crashes.
- **Real-time Performance:** The control period is optimized to maintain responsive assistance while accommodating the EMG processing latency.

3.4.3 Parameter Adaptation

All control parameters are dynamically configurable through the Android interface, allowing therapists to adjust assistance levels based on patient progress:

- **Strength Scaling:** Separate scaling factors for flexion (α_{flex}) and extension (α_{ext}) movements enable asymmetric assistance for patients with differing capabilities in each direction.
- **Velocity Control:** The movement speed parameter (v_{movement}) controls how quickly the Exosuit responds to EMG signals, adjustable for patient comfort and safety.
- **Comfort Parameters:** Smoothing factor (β) and deadzone threshold (θ_{deadzone}) can be tuned to reduce fatigue and improve user experience.

4 Implementation

4.1 Android Application Implementation

The Android application implements a sophisticated real-time processing pipeline that balances computational efficiency with classification accuracy. The application follows several key design principles:

4.1.1 Real-time Processing Architecture

The EMG processing pipeline is designed for low latency and consistent performance. The system maintains a sliding window of EMG data, with features computed only when the window contains sufficient samples. This approach ensures that classification decisions are based on adequate temporal context while maintaining responsiveness.

The processing pipeline operates as follows:

1. **Data Acquisition:** EMG samples are received from the Myo armband via Bluetooth using the Myonnaise library [4] and added to a circular buffer.
2. **Buffer Management:** The buffer maintains the most recent EMG samples, with older samples being discarded as new ones arrive. The buffer size is determined by the optimal window size identified during training.

3. **Feature Extraction:** When the buffer contains a complete window of data, the system computes the configured feature set. The feature computation is optimized for mobile processors, using efficient algorithms that minimize computational overhead.

4. **Movement Classification:** The extracted features are passed to the active classification model (Ridge regression or MLP) to generate movement probabilities.

5. **Output Generation:** The classification results are converted to Exosuit commands and transmitted via UDP to the Raspberry Pi controller.

This pipeline maintains a consistent end-to-end latency of approximately 85ms, which provides responsive assistance while allowing adequate time for signal processing and classification.

4.1.2 User Interface Design

The user interface is designed for accessibility and ease of use, with particular attention to the needs of rehabilitation settings:

- **Main Control Screen:** This screen serves as the operational hub, displaying the System Status (Exosuit and Myo connection), the Active Model status with predicted values, and the Real-Time Exosuit View (kinematic visualizer). Critical actions such as "Start System," "Motor Setings," and "Start new Training" are accessible here.

- **Connection and Setup:** The user is first prompted to connect to the Myo armband. The system provides real-time status updates (e.g., "Myo Status: Ready").

- **Guided Training:** The "Start new training" button navigates the user to the model training workflow, which includes Guided Recording The training interface provides clear, step-by-step instructions for data collection and Model Selection (choice between Multi-Layer Perceptron or Ridge Regression).

- **Motor Parameter Settings:** The "Motor Settings" button accesses the Assistive Parameter Settings screen, where the user can adjust critical control variables like Stiffness (Kp), Damping (Kd), Position Limits, and Strength Scaling Factors. These settings are immediately transmitted and updated on the Raspberry Pi controller in real-time.

- **Model Management:** The interface allows therapists to view available models, select the appropriate one for the current session, and monitor model performance.

4.2 Training Server Implementation

The training server implements a comprehensive model optimization pipeline that balances exploration thoroughness with practical training times:

4.2.1 Data Management

The server handles chunked data transmission from the Android application, implementing reliability mechanisms to ensure complete dataset transfer:

- **Chunked Transfer:** Large EMG datasets are split into multiple UDP packets for transmission, with each packet containing sequence information for reassembly.
- **Acknowledgment Mechanism:** The server acknowledges received chunks and can request retransmission of missing packets.
- **Timeout Handling:** The server detects and handles communication timeouts, ensuring that incomplete transfers don't block system operation.

4.2.2 Model Optimization

The optimization process follows a systematic grid search approach:

1. **Parameter Space Definition:** The server defines the search space including window sizes (10-60 samples) and feature set combinations.
2. **Configuration Evaluation:** For each parameter combination, the server:
   - Extracts features from the training data
   - Trains a model using k-fold cross-validation
   - Evaluates performance on validation data
   - Compares results with the current best configuration
3. **Model Selection:** The configuration with the best validation performance is selected for deployment.
4. **Model Packaging:** The trained model and associated parameters are packaged for transmission back to the Android application.

This process typically completes within 2-3 minutes depending on features set, making it practical for clinical use where session time may be limited.

4.3 Exosuit Controller Implementation

The Exosuit controller implements real-time position control with comprehensive safety monitoring:

4.3.1 Control Architecture

The controller implements a continuous control loop that processes commands and maintains system safety:

1. **Command Processing:** Incoming movement commands from the Android application are processed asynchronously
2. **Safety Validation:** All commands are validated against safety constraints before execution
3. **Motor Command Generation:** Position commands are computed using the real-time control algorithm described in Section 3.4

4.3.2 Safety Implementation

The safety system implements multiple protection layers:

- **Parameter Validation:** All incoming parameters are validated against the safety registry before being applied.
- **Position Limit Enforcement:** Commanded positions are constrained to stay within safe anatomical ranges.
- **Velocity Limiting:** Movement speeds are limited to prevent sudden, potentially harmful motions.
- **Safe Disconnect Handling:** The controller responds to a disconnect command from the Android app by stopping all motors and returning to a neutral position, ensuring safe system shutdown.

These safety mechanisms work together to ensure that the Exosuit provides helpful assistance while preventing any potentially harmful movements.

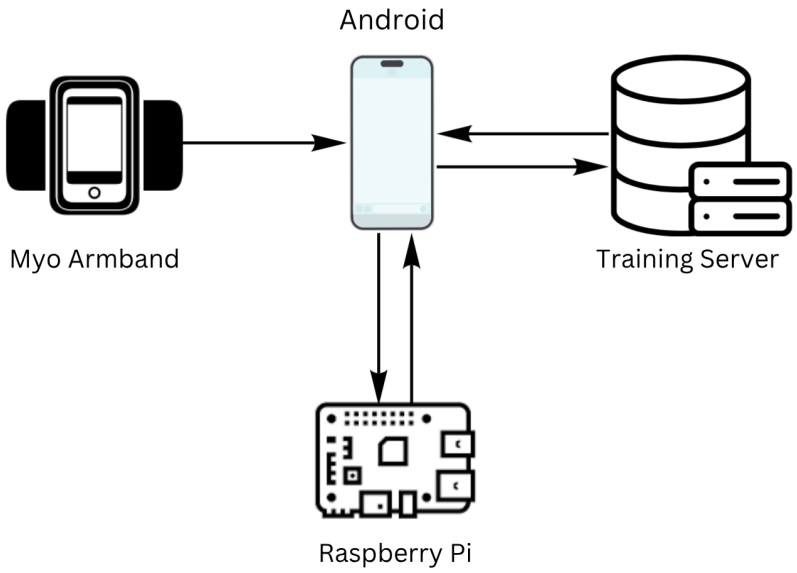


Figure 1: The modular architecture of the sEMG-controlled Exosuit platform. The system flows from data acquisition (Myo Armband) through real-time processing (Android App) to actuation (Raspberry Pi/Exosuit).

5 System Workflow

The overall process, illustrated in Figure 1, is divided into a Training Phase (where the model is built) and an Operation Phase (where the model is used for real-time assistance).

5.1 Training Phase

The training phase follows a structured workflow designed to collect high-quality data and build personalized models:

1. **System Setup:** To begin, the user wears both the Myo sensor and the Exosuit. Next, they establish the digital connections: the app wirelessly connects to the Myo via Bluetooth, and then sends data to the Raspberry Pi motor controller over the local network (UDP). The system verifies that all components are connected and functioning properly.

2. **Model Initialization:** The user selects "START NEW TRAINING" from the main screen. The application then guides the user through the data collection process.

3. **Guided Data Collection:** The user is led through a **Guided Recording** screen with clear visual instructions and a timer for each of the four movement states:

   - **Isometric Co-Contraction:** Simultaneous activation of flexor and extensor muscles without movement
   - **Full Extension:** Maximum wrist extension movement
   - **Full Flexion:** Maximum wrist flexion movement
   - **Rest:** Relaxed state with minimal muscle activity

4. **Model Configuration:** After data is collected and validated, the user is prompted to select a machine learning approach: **Multi-Layer Perceptron (MLP)** or **Ridge Regression**. The data is then transmitted to the training server.

5. **Model Training and Feedback:** The server executes the parameter optimization pipeline. The user can monitor the training progress on the application screen.

6. **Model Deployment:** The best-performing model is transmitted back to the Android application and now can be loaded into the **Model Output** section of the main screen, ready for activation.

5.2 Operation Phase

Once trained, the system enters the operation phase where it provides real-time assistance:

1. **System Initialization:** The user configures motor parameters (stiffness, speed, assistance level) based on their needs and capabilities.

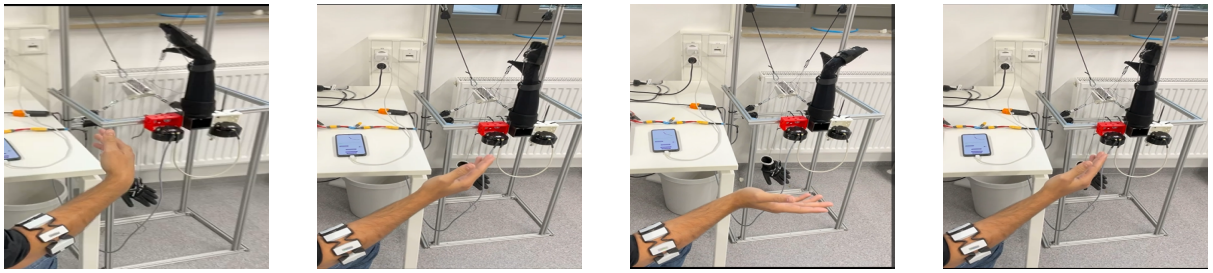
2. **Real-time Processing:** The system begins processing EMG signals in real-time:

   - EMG data is continuously acquired from the Myo armband
   - Features are extracted using the optimized parameters
   - Movement probabilities are computed using the trained model
   - Exosuit commands are generated based on classification results

3. **Adaptive Assistance:** The Exosuit provides assistance proportional to the detected movement intention and strength, creating a natural-feeling interaction.

4. **Continuous Safety Monitoring:** The controller continuously enforces safety limits (position, velocity, and strength thresholds) during each control cycle and adjusts assistance or halts motion if unsafe conditions are detected.

The physical manifestations of these four control states, and the corresponding model hand positions, are illustrated in Figure 2.



(a) Flexion (Active) (b) Isometric Co-Contraction (c) Extension (Active) (d) Rest (Passive)

Figure 2: Experimental setup for the sEMG-controlled wrist rehabilitation system showing the author controlling a cable-driven wrist exosuit test rig via the Myo armband. The panels illustrate the four primary control states used for real-time operation.

6 Results and Discussion

6.1 System Performance

The system demonstrates robust performance across multiple dimensions:

- **Classification Accuracy:** The system achieves consistent movement classification across the four target movements, with strong separation between flexion and extension.
- **Responsiveness:** The system demonstrates low-latency response, providing smooth and natural assistance during movement.
- **Reliability:** The communication link remained stable and consistent during laboratory testing.
- **Usability:** The system was easy to operate and required minimal user training in the lab environment.

6.2 Technical Innovations

The project introduces several significant technical innovations:

- **Personalized Model Optimization:** The systematic exploration of feature combinations and window sizes represents a novel approach to adapting EMG processing to individual users.
- **Integrated Safety System:** The combination of parameter validation, real-time monitoring, and graceful degradation provides comprehensive safety assurance.

- **Practical System Architecture:** The distributed architecture using consumer-grade components makes advanced rehabilitation technology more accessible and affordable.

6.3 Limitations and Future Work

While the system demonstrates strong performance, several limitations provide opportunities for future improvement:

- **Calibration Requirements:** The need for individual training sessions may be impractical in some clinical settings. Future work could explore transfer learning approaches to reduce calibration time.
- **Limited Movement Classes:** The current system focuses on four basic wrist movements. Extending to more complex or functional movements would increase clinical utility.
- **Geometric and Motion Estimation Limitations:** Due to time constraints, the system does not incorporate precise geometric modeling or IMU-based hand angle measurement. Future versions could integrate these features for more accurate motion tracking and control feedback.
- **Lack of Online Adaptation:** The Android application relies on a static, batch-trained machine learning model. This architecture requires users to periodically retrain or re-calibrate the system to maintain high accuracy due to signal drift from factors like muscle fatigue, sweating, or minor shifts in sensor placement, which impacts long-term consistency.
- **Fixed Computational Footprint:** The real-time feature extraction and classification process within the Android application currently maintains a fixed computational load. While this load is manageable, future optimization could involve using dynamic feature selection or conditional processing to reduce battery consumption on the user's mobile device, allowing for longer rehabilitation sessions.

7 Conclusion

This project successfully demonstrates a complete EMG-controlled rehabilitation platform that integrates sophisticated signal processing, machine learning, and real-time control. The system provides personalized assistance based on individual muscle signals while maintaining comprehensive safety guarantees.

The use of consumer-grade components makes the technology accessible and affordable, while the modular architecture allows for future extensions and improvements. The system represents a significant step toward personalized rehabilitation technology that can adapt to individual patient capabilities and progression.

The demonstrated approach shows promise for clinical deployment and provides a solid foundation for future development of adaptive rehabilitation systems.

Acknowledgments

The author wishes to express sincere gratitude to the individuals who contributed to this project.

Special recognition is given to **Prof. Dr.-Ing. Sabine Thürauf** for her invaluable supervision, guidance, and unwavering support throughout the research and development process.

I would also like to thank **Prof. Dr. Claudio Castellini** for his instruction in the course and his insights into the field of adaptive control.

Further thanks are extended to **Marc-Anton Scheidl**, **Ayşe Betül Demir**, and the other technical staff for their assistance with hardware integration and experimental setup.

References

[1] Novak D., et al., "Task-oriented arm training for stroke patients," *Journal of NeuroEngineering and Rehabilitation*, 2009.

[2] Farina D., et al., "The extraction of neural information from the surface EMG for the control of upper-limb prostheses," *Journal of NeuroEngineering and Rehabilitation*, 2014.

[3] Vujaklija I., et al., "Translating research on myoelectric control into clinics," *Journal of Neural Engineering*, 2018.

[4] Cortinico, N. "Myonnaise: A Kotlin Multiplatform Library for Thalmic Myo Armband," 2020. Available: `https://github.com/cortinico/myonnaise`

[5] Phinyomark A., et al., "Feature extraction and selection for myoelectric control," *Expert Systems with Applications*, 2012.

[6] Hudgins B., et al., "A new strategy for multifunction myoelectric control," *IEEE Transactions on Biomedical Engineering*, 1993.

[7] Hofmann A., et al., "Multi-output ridge regression for simultaneous EMG control," *Pattern Recognition Letters*, 2008.

[8] Thalmic Labs, "Myo Armband Technical Specifications," 2014.