DIY Muscle Classification with Machine Learning

Arya Nalavade, Fatima Shaikh, Daniel Melvin

Background

Electromyography (EMG) is a technique that measures the electrical activity produced by skeletal muscles during contraction [1]. Surface EMG (sEMG) is widely used in research and clinical applications due to its non-invasive nature and its ability to provide insight into muscle activation patterns [2]. When a muscle contracts, it generates an electrical signal, which can be detected through electrodes placed on the skin [1]. These signals can then be processed and classified to understand which muscles are active and infer the corresponding grips.

Hand grips are a critical function of the human hand, allowing for gross motor movements, such as lifting heavy objects, and fine motor tasks, such as writing. Classifying different types of grips based on EMG signals can be used in prosthetics, rehabilitation, and human-computer interaction [3]. For this project we focused on the six basic grips that represent a wide range of hand functions. The power grip, which is a fist, uses the flexor digitorum profundus (FDP) and flexor digitorum superficialis (FDS), allowing for stronger finger flexion [4]. The spherical grip is used to grasp round objects and engages the muscles along the FDS and FDP [5]. The cylindrical grip activates both of these muscles around cylindrical objects. The hook grip involves flexion at the proximal and distal interphalangeal joints without the use of the thumb [6]. A lateral pinch utilizes the first dorsal interosseous (FDI) and flexor pollicis longus (FPL) to press objects between the thumb and side of the index finger [7]. Lastly, the tripod pinch engages the FDI and FPL to hold objects between the thumb, index and middle fingers [8].

Solution

The objective of this project is to design and build a wearable muscle activity sensor that can reliably detect different hand grips. The original plan was to design a custom circuit

involving op amps, along with high pass and low pass filters, to process the raw EMG signals before sending them to the microcontroller for analysis. However, as we progressed, we realized that designing and tuning the circuit introduced noise issues. Upon conducting a more thorough literature search, the approach shifted to using prebuilt EMG modules for a more streamlined and reliable process. The MyoWare 2.0 Muscle sensor was chosen as it integrated all the circuitry internally. By using two MyoWare sensors placed on the forearm, distinct EMG signal patterns were collected, corresponding to different grip types. One sensor was placed on the medial side over the flexor muscles and the other sense over the lateral side of the flexor muscles (Figure 1). This positioning allowed us to capture distinct EMG patterns corresponding to each of the six targeted hand grips.

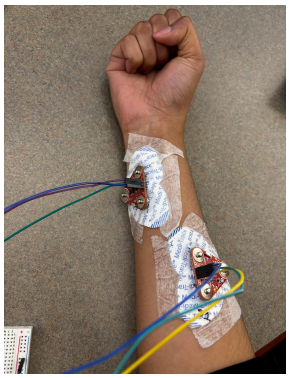

Figure 1: Muscle sensor placement on forearm

System Hardware and Setup

Our hardware setup included two MyoWare 2.0 Muscle Sensors and an Arduino Leonardo microcontroller. The MyoWare sensors are compact EMG devices with built-in signal rectification, envelope detection, and gain control, enabling them to output smooth, amplified analog signals corresponding to muscle activation levels [9]. These signals range from 0V (relaxed) to 5V (fully contracted). The ENV pins were connected to the Arduino's analog inputs,

which sampled the data and transmitted it to a laptop over serial communication at a baud rate of 9600.

Electrode placement was essential for capturing clean, distinguishable EMG signals for grip classification. One sensor was placed on the upper medial forearm to capture finger flexion activity, while the second sensor was positioned on the lower lateral forearm to better detect finer thumb and finger movements involved in precision and lateral grips. To minimize noise and motion artifacts, the skin was lightly cleaned and electrodes were securely attached [10]. The MyoWare sensors' compact design and adhesive backing allowed for easy and consistent placement across multiple trials.

Data Acquisition

To build a robust training dataset for the machine learning model, we systematically recorded EMG signals during six distinct grip types: power grip, spherical grip, cylindrical grip, hook grip, lateral grip, and pinch grip. These grips were selected because they involve distinct muscle activation patterns that could be differentiated using our dual-sensor electrode setup. Each grip type was performed under carefully controlled conditions to ensure consistency and minimize noise during data collection.

During each data collection session, the subject firmly held a specific grip for 90 seconds while maintaining a steady, moderate contraction intensity. During this period, the Arduino Leonardo read in analog readings from two MyoWare sensors to a laptop via serial communication, capturing paired values from the upper medial forearm (Sensor 1) and lower lateral forearm (Sensor 2). The data was saved into labeled CSV files for later processing. Holding the grip for an extended period allowed the EMG signals to stabilize and provided enough data for multiple non-overlapping analysis windows. To minimize fatigue effects and

preserve signal quality, the subject maintained consistent effort and rested 1–2 minutes between trials. This systematic procedure produced a balanced, labeled dataset suitable for feature extraction, model training, and evaluation [10].

Data Preprocessing

Following data collection, several preprocessing steps were applied to prepare the dataset for effective machine learning. First, we ensured that each grip class contributed approximately the same number of samples to prevent the model from becoming biased toward any specific grip. Next, outlier removal was performed using empirical thresholds to filter out extreme values caused by motion artifacts, electrode shifts, or inconsistent contractions. After that, we standardized the signals through z-score normalization to remove differences in signal amplitude between sessions and individuals. Preprocessing was essential to improve the model's ability to learn general patterns rather than memorizing noise or session-specific variations. Without careful preprocessing, inconsistencies in signal strength, recording artifacts, and class imbalance could degrade the classifier's performance, resulting in overfitting or poor generalization [11]. By balancing, cleaning, and normalizing the data before feature extraction, we ensured that the machine learning model could focus purely on the meaningful structure of the EMG signals related to each grip type.

Feature Extraction and Feature Importance

After collecting raw EMG signals from both sensors, feature extraction was applied to convert the continuous time-series data into structured input for machine learning. The EMG recordings were divided into windows of 50 samples, with each window treated as a separate observation. From each window, seven key time-domain features were extracted from each

sensor: Root Mean Square (RMS), Mean Absolute Value (MAV), Waveform Length (WL), Zero Crossings (ZC), Slope Sign Changes (SSC), Variance (VAR), and Skewness (SKEW). These features summarized important signal characteristics: RMS estimated signal energy, MAV captured average amplitude, WL measured cumulative complexity, ZC counted oscillations, SSC indicated slope transitions, Variance quantified signal spread, and Skewness measured asymmetry in the distribution [11].

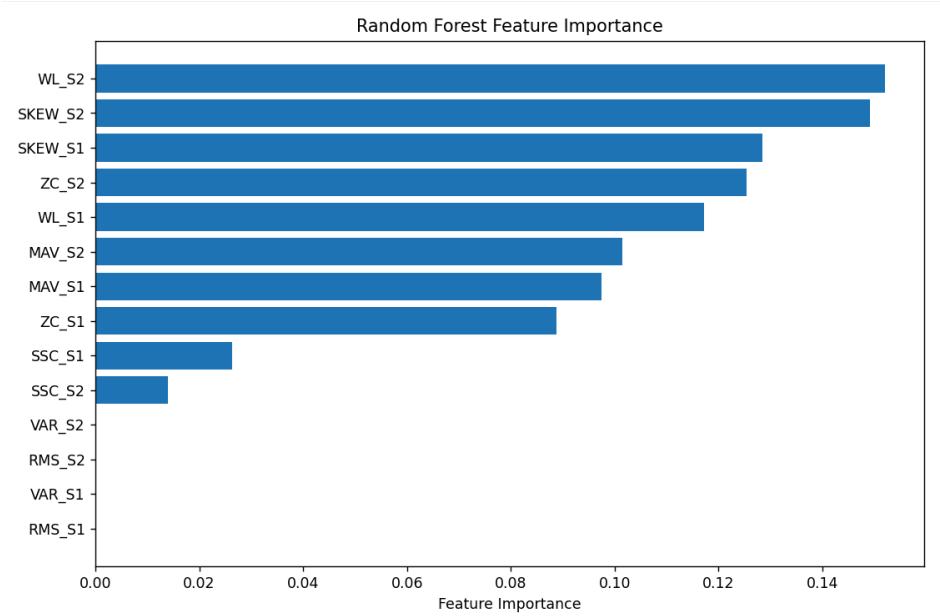

Figure 2: Random Forest feature importance rankings for EMG-based grip classification

Each 50-sample window produced a 14-dimensional feature vector (seven features from each sensor), forming the input dataset X, while the corresponding grip type formed the output dataset y. These extracted features summarized complex EMG patterns into numerical representations suitable for machine learning [11]. A Random Forest classifier was trained using these feature vectors, configured with 200 trees and a maximum depth of 10 to balance complexity and avoid overfitting. Random Forest also ranked feature importance based on how much each feature improved decision splits. As shown in Figure 2, Waveform Length (WL) and Skewness (SKEW) from Sensor 2 were the most influential, followed by Skewness and Zero

Crossings (ZC) from Sensor 1. Simpler features like RMS and MAV contributed less, emphasizing that signal complexity and dynamic transitions were stronger indicators of grip type than raw amplitude alone [11].

Machine Learning Model Selection

For grip classification based on EMG signals, we selected a Random Forest classifier due to its robustness, flexibility, and strong performance on moderate-sized datasets. Random Forest builds multiple decision trees and combines their predictions, reducing the risk of overfitting while capturing complex, nonlinear relationships between EMG features and grip types.

Our model was configured with 200 trees to ensure diversity and stability, and a maximum depth of 10 to control complexity and avoid overfitting. This setup balanced the ability to recognize important patterns without memorizing noise. Random Forest was particularly suited to this project because it handles multi-class problems naturally, requires minimal assumptions about feature distributions, and provides feature importance rankings (as shown in Figure 2), helping to interpret which EMG features contributed most to grip classification. Its generalization ability on small, noisy datasets made it ideal for real-time EMG applications [11].

Machine Learning Model Training

To evaluate the model's performance and ensure generalization to unseen data, the full dataset was split into a training set (80%) and a testing set (20%). This separation allowed the model to learn patterns from the training data while preserving an independent testing set to simulate real-world conditions. Stratified sampling was used to maintain equal proportions of

each grip class (lateral, spherical, power, pinch, cylindrical, and hook) in both sets, preventing bias and ensuring balanced learning across all grips [10].

During training, the Random Forest classifier was fed feature vectors (X_train), each containing 14 extracted features (RMS, MAV, WL, ZC, SSC, VAR, and SKEW) from the two sensors. The corresponding grip labels (y_train) guided the model in linking EMG patterns to specific grips. The Random Forest built 200 decision trees, each trained on random subsets of data, and learned decision rules for classification. A maximum tree depth of 10 was set to prevent overfitting, encouraging the model to capture generalizable trends rather than memorizing noise [10].

Machine Learning Model Evaluation

	precision	recall	f1-score	support
cylindrical	1.00	0.71	0.83	7
hook	1.00	1.00	1.00	7
lateral	0.64	1.00	0.78	7
pinch	0.80	0.57	0.67	7
power	1.00	0.71	0.83	7
spherical	0.80	1.00	0.89	8
accuracy			0.84	43
macro avg	0.87	0.83	0.83	43
weighted avg	0.87	0.84	0.83	43

Figure 3: Machine Learning Model Classification Report

The performance of the trained Random Forest classifier was evaluated using the independent testing set, comprising 20% of the total dataset. Key metrics—including accuracy, precision, recall, F1-score, and a confusion matrix—were analyzed to assess the model's generalization ability. As shown in Figure 3: Classification Report, the classifier achieved an overall accuracy of 84%, meaning 84% of the grip predictions on unseen data were correct.

Precision, which measures the proportion of correct positive predictions, was highest (1.00) for the cylindrical, hook, and power grips, while the lateral grip showed a lower precision of 0.64, indicating occasional misclassifications. Recall, the proportion of actual grips correctly identified, was perfect (1.00) for the hook, lateral, and spherical grips, with slightly lower recall values for cylindrical (0.71), pinch (0.57), and power (0.71) grips. F1-scores, which balance precision and recall, ranged from 0.67 (pinch) to 0.89 (spherical), with most grips scoring above 0.75. Support, representing the number of test samples per class (7–8 samples each), was evenly distributed, strengthening the reliability of macro-averaged results.

Overall, the macro-averaged precision, recall, and F1-score were 87%, 83%, and 83%, respectively, highlighting the model's robust and balanced performance across different grip types, although minor sensitivity issues were noted for the pinch grip.

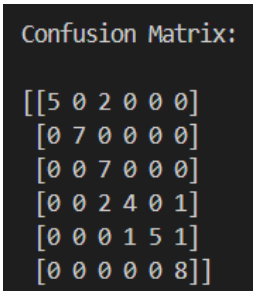

Figure 4: Machine Learning Model Confusion Matrix

The confusion matrix presented in Figure 4 illustrates the distribution of correct and incorrect grip classifications. Most predictions were concentrated along the diagonal, indicating that the model correctly classified the majority of test samples. The hook, spherical, and lateral grips achieved perfect classification with no misclassifications, reflecting their distinct EMG activation patterns. Occasional misclassifications occurred between the cylindrical, pinch, and power grips, where overlapping muscle activity likely caused confusion. For example, some cylindrical grips were misclassified as pinch grips, and some power grips were predicted as

cylindrical. Despite these overlaps, the overall low rate of off-diagonal errors demonstrates that the model reliably distinguished most grips, with only minor confusion among similar patterns.

Results

```
Raw Avg Sensor1: 1.94, Sensor2: 929.12
Matched Grip: SPHERICAL

Raw Avg Sensor1: 1.96, Sensor2: 929.04
Matched Grip: SPHERICAL

Raw Avg Sensor1: 1.88, Sensor2: 867.78
Matched Grip: SPHERICAL
```

Figure 5: Results showing a spherical grip

```
Raw Avg Sensor1: 1.82, Sensor2: 113.02
Matched Grip: LATERAL

Raw Avg Sensor1: 1.84, Sensor2: 56.08
Matched Grip: LATERAL

Raw Avg Sensor1: 1.88, Sensor2: 54.30
Matched Grip: LATERAL
```

Figure 6: Results showing a lateral grip

```
Raw Avg Sensor1: 301.82, Sensor2: 105.14
Raw Avg Sensor1: 753.56, Sensor2: 82.92
Matched Grip: CYLINDRICAL

Raw Avg Sensor1: 904.52, Sensor2: 58.88
Raw Avg Sensor1: 499.78, Sensor2: 152.56
Matched Grip: CYLINDRICAL
```

Figure 7: Results showing a cylindrical grip

```
Raw Avg Sensor1: 624.56, Sensor2: 889.16
Matched Grip: POWER

Raw Avg Sensor1: 811.90, Sensor2: 929.20
Matched Grip: POWER

Raw Avg Sensor1: 742.00, Sensor2: 929.20
Matched Grip: POWER
```

Figure 8: Results showing a power grip

The trained grip classification program successfully identified grip types in real time based on distinct EMG patterns from two sensors. As shown in Figures 5–8, the model matched live sensor values to their corresponding grip categories if the confidence is above 85%. Figure 5 shows accurate detection of the spherical grip, where Sensor 1 values were low (~1.9) and Sensor 2 values were high (~900+), aligning with expected patterns. Figure 6 displays results for the lateral grip, with both sensors showing low activity, matching its known EMG profile. In Figure 7, the model correctly classified the cylindrical grip across varying Sensor 1 levels, while Sensor 2 remained within the expected range (50–150). Figure 8 highlights the power grip, where both sensors showed strong activation, particularly Sensor 2 readings above 880,

confirming the model's ability to detect high-intensity grips. Overall, these examples demonstrate the model's ability to generalize to live data and reliably identify different grips based on their EMG signatures.

Contributions

Arya was responsible for developing the machine learning model. She extracted key features from the EMG signal data, built the model to read in these extracted features, trained the classifier, and wrote the final code used to predict the hand grip based on live EMG inputs. Fatima worked on determining optimal electrode placement on the forearm. She worked on the testing and data acquisition phase, collecting EMG signals for the different grips, and worked on preprocessing the data, including identifying and removing outliers that could negatively impact model performance. Daniel contributed by writing the Arduino code necessary to connect with the sensors and stream the EMG signal data. He also conducted an additional literature search on machine learning techniques to help refine the project approach. Although each team member had specific responsibilities, all members collaborated closely to ensure the success of the overall system.

References

[1] C. J. D. Luca, “The Use of Surface Electromyography in Biomechanics,” Journal of Applied Biomechanics, vol. 13, no. 2, pp. 135–163, May 1997, doi: 10.1123/jab.13.2.135.

[2] D. Farina and R. M. Enoka, “Evolution of surface electromyography: From muscle electrophysiology towards neural recording and interfacing,” Journal of Electromyography and Kinesiology, vol. 71, p. 102796, Aug. 2023, doi: 10.1016/j.jelekin.2023.102796.

[3] F. S. Miften, M. Diykh, S. Abdulla, S. Siuly, J. H. Green, and R. C. Deo, “A new framework for classification of multi-category hand grasps using EMG signals,” Artificial Intelligence in Medicine, vol. 112, p. 102005, Feb. 2021, doi: 10.1016/j.artmed.2020.102005.

[4] B. E. Lung and B. Burns, “Anatomy, Shoulder and Upper Limb, Hand Flexor Digitorum Profundus Muscle,” in StatPearls, Treasure Island (FL): StatPearls Publishing, 2025. Accessed: Apr. 28, 2025. [Online]. Available: http://www.ncbi.nlm.nih.gov/books/NBK526046/

[5] “BIOMECHANICS OF THE HAND.” Accessed: Apr. 28, 2025. [Online]. Available: https://ouhsc.edu/bserdac/dthompso/web/namics/hand.htm

[6] T. Bayer, A. Schweizer, M. Müller-Gerbl, and G. Bongartz, “Proximal Interphalangeal Joint Volar Plate Configuration in the Crimp Grip Position,” The Journal of Hand Surgery, vol. 37, no. 5, pp. 899–905, May 2012, doi: 10.1016/j.jhsa.2012.02.016.

[7] D. Kaiser, A.-C. Masquelet, A. Sautet, and A. Cambon-Binder, “Reconstruction of lateral pinch in an isolated paralysis of the first dorsal interosseous muscle - A new surgical technique,” Orthop Traumatol Surg Res, vol. 106, no. 2, pp. 353–356, Apr. 2020, doi: 10.1016/j.otsr.2019.08.024.

[8] A. L. Ladd, “The Teleology of the Thumb: on Purpose and Design,” J Hand Surg Am, vol. 43, no. 3, pp. 248–259, Mar. 2018, doi: 10.1016/j.jhsa.2018.01.002.

[9] Advancer Technologies, “MyoWare 2.0 Muscle Sensor Datasheet,” Accessed: Apr. 2025. [Online]. Available: https://myoware.com/products/muscle-sensor/

[10] L. J. Hargrove, K. Englehart, and B. Hudgins, “A training strategy to reduce classification degradation due to electrode displacements in pattern recognition based myoelectric control,” *Biomedical Signal Processing and Control*, vol. 3, no. 2, pp. 175–180, 2008.

[11] M. Oskoei and H. Hu, “Support vector machine-based classification scheme for myoelectric control applied to upper limb,” *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 8, pp. 1956–1965, 2008.

Appendix

Arduino Code

C/C++

```
void setup() {
  Serial.begin(19200);
  while (!Serial);
  Serial.println("Sensor Reading...");
}

void loop() {
  int sensor1 = analogRead(A0); // Lower lateral forearm
  int sensor2 = analogRead(A1); // Upper medial forearm

  Serial.print(sensor1);
  Serial.print(",");
  Serial.println(sensor2);

  delay(50); // 20 readings per second
}
```

Data Acquisition Code

Python

```
import serial
import csv
import time

# --- SETUP ---
port = "COM3"  # <-- Replace with your Arduino's port
baud = 9600
output_file = "emg_data.csv"
grip_label = "power"  # <-- Change this for each grip trial
```

```
# --- CONNECT TO SERIAL ---
ser = serial.Serial(port, baud)
print(f"Connected to {port}")
time.sleep(2)  # wait for serial to initialize

# --- CSV SETUP ---
with open(output_file, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["Time", "Sensor1", "Sensor2", "Label"])  # header

    print("Logging... Press CTRL+C to stop")

    try:
        while True:
            line = ser.readline().decode('utf-8').strip()
            parts = line.split(',')
            if len(parts) == 2:
                timestamp = time.time()
                sensor1 = int(parts[0])
                sensor2 = int(parts[1])
                writer.writerow([timestamp, sensor1, sensor2, grip_label])
                print(timestamp, sensor1, sensor2, grip_label)
    except KeyboardInterrupt:
        print("Logging stopped.")

ser.close()
```

Machine Learning Model Code

Python

```
# === Final Optimized Grip Classifier Training Script (Improved Accuracy) ===

import pandas as pd
import numpy as np
import os
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, StratifiedKFold,
cross_val_score
from sklearn.metrics import classification_report, confusion_matrix
```

```
import joblib
import matplotlib.pyplot as plt

# === Step 1: Load and combine labeled grip data files ===
file_paths = {
    "lateral": "lateralFINAL.csv",
    "spherical": "sphericalFINAL.csv",
    "power": "powerFINAL.csv",
    "pinch": "pinchFINAL.csv",
    "cylindrical": "cylindricalFINAL.csv",
    "hook": "hookFINAL.csv"
}

data = []
for label, file in file_paths.items():
    df = pd.read_csv(file)
    df["label"] = label
    data.append(df)

combined_df = pd.concat(data, ignore_index=True)

# === Step 2: Feature extraction ===
def extract_features(df, window_size=50, zc_thresh=8, ssc_thresh=8):
    features = []
    labels = []

    for label in df['label'].unique():
        grip_df = df[df['label'] == label][['Sensor1',
'Sensor2']].values.astype(float)
        for i in range(0, len(grip_df) - window_size, window_size):
            window = grip_df[i:i + window_size]
            feature_vector = []

            for col in range(window.shape[1]):
                signal = window[:, col]
                std = np.std(signal)
                signal = (signal - np.mean(signal)) / std if std != 0 else
signal - np.mean(signal)

                rms = np.sqrt(np.mean(signal**2))
                mav = np.mean(np.abs(signal))
                wl = np.sum(np.abs(np.diff(signal)))
                zc = np.sum(np.diff(np.signbit(signal)) != 0)
                diff1 = np.diff(signal[:-1])
```

```
                diff2 = np.diff(signal[1:])
                ssc = np.sum((diff1 * diff2 < 0) & (np.abs(diff1 - diff2) >
ssc_thresh))

                var = np.var(signal)
                skew = np.mean((signal - np.mean(signal))**3) /
(np.std(signal)**3 + 1e-6)

                feature_vector.extend([rms, mav, wl, zc, ssc, var, skew])

            features.append(feature_vector)
            labels.append(label)

    return np.array(features), np.array(labels)

# Extract features and split data
X, y = extract_features(combined_df)
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
test_size=0.2, random_state=42)

# Train Random Forest Classifier with tuned parameters
clf = RandomForestClassifier(n_estimators=200, max_depth=10,
class_weight='balanced', random_state=42)
clf.fit(X_train, y_train)

# Evaluate Model
y_pred = clf.predict(X_test)
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n")
print(confusion_matrix(y_test, y_pred))

# === Step 6: Feature Importance Visualization ===
feature_labels = [
    'RMS_S1', 'MAV_S1', 'WL_S1', 'ZC_S1', 'SSC_S1', 'VAR_S1', 'SKEW_S1',
    'RMS_S2', 'MAV_S2', 'WL_S2', 'ZC_S2', 'SSC_S2', 'VAR_S2', 'SKEW_S2'
]
importances = clf.feature_importances_
sorted_idx = np.argsort(importances)

plt.figure(figsize=(9, 6))
plt.barh(range(len(importances)), importances[sorted_idx], align='center')
plt.yticks(range(len(importances)), np.array(feature_labels)[sorted_idx])
plt.xlabel("Feature Importance")
```

```
plt.title("Random Forest Feature Importance")
plt.tight_layout()
plt.show()

# === Step 7: Save the trained model ===
joblib.dump(clf, "grip_classifier.pkl")
print("\nModel saved as grip_classifier.pkl")
```

Classification Code

Python

```python
import serial
import numpy as np
import joblib
import time

# Load trained ML model
clf = joblib.load("grip_classifier.pkl")  # Load the pre-trained Random Forest classifier

# Serial connection setup
ser = serial.Serial('COM3', 9600)  # Connect to Arduino (Update COM port if needed)
time.sleep(2)  # Allow time for Arduino to reset after opening the serial port

# Initialize data buffers
window_size = 50  # Number of EMG samples to collect before making a prediction
buffer1 = []
buffer2 = []

print("Listening for EMG... Press Ctrl+C to stop.\n")

while True:
    try:
        # Read and parse serial input
        line = ser.readline().decode(errors='ignore').strip()  # Read and decode a line of serial data
        parts = line.split(',')
        if len(parts) != 2:
```

```
            continue

        emg1 = float(parts[0])  # Sensor 1 reading
        emg2 = float(parts[1])  # Sensor 2 reading
        buffer1.append(emg1)
        buffer2.append(emg2)

        # When enough samples are collected, predict grip
        if len(buffer1) == window_size:
            features = []  # Will hold features extracted from the 50-sample
window

            # Extract features for each sensor
            for signal in [buffer1, buffer2]:
                signal = np.array(signal)
                std = np.std(signal)
                # Normalize signal (z-score) to remove amplitude shifts
                signal = (signal - np.mean(signal)) / std if std != 0 else
signal - np.mean(signal)

                # Time-domain feature calculations
                rms = np.sqrt(np.mean(signal**2))           # Root Mean Square
                mav = np.mean(np.abs(signal))            # Mean Absolute Value
                wl = np.sum(np.abs(np.diff(signal))) #Waveform Length crossings
                zc = np.sum(np.diff(np.signbit(signal)) != 0)    # Zero
                diff1 = np.diff(signal[:-1])
                diff2 = np.diff(signal[1:])
                ssc = np.sum((diff1 * diff2 < 0) & (np.abs(diff1 - diff2) > 5))
# Slope Sign Changes

                # Add extracted features to feature vector
                features.extend([rms, mav, wl, zc, ssc])

            features = np.array(features).reshape(1, -1)  # Reshape model input

            # Predict grip type using ML model
            prediction = clf.predict(features)[0]# Predicted label
            proba = clf.predict_proba(features)            # Probabilities
            confidence = np.max(proba)      # Confidence of the prediction

            # Print prediction results
            print(f"Predicted Grip: {prediction} (Confidence:
{confidence:.2f})")
```

```
            # Print average raw signal values for debugging
            avg_s1 = np.mean(buffer1)
            avg_s2 = np.mean(buffer2)
            print(f"Raw Avg Sensor1: {avg_s1:.2f}, Sensor2: {avg_s2:.2f}\n")

            #
            buffer1.clear()
            buffer2.clear()

    except KeyboardInterrupt:
        print("\nStopped by user.")
        break
    except Exception as e:
        print("Error:", e)
```