Smart Prosthetic Hand

P. Teja Prakash
CB.AI.U4AIM24136
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu, India
cb.ai.u4aim24136@cb.students.amrita.edu

S.V.S Ankith
CB.AI.U4AIM24147
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu, India
cb.ai.u4aim24147@cb.students.amrita.edu

A. Guru Jaya Surya Yadav
CB.AI.U4AIM24101
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu, India
cb.ai.u4aim24101@cb.students.amrita.edu

J. Tej Krishna Sai
CB.AI.U4AIM24117
School of Artificial Intelligence
Amrita Vishwa Vidyapeetham
Coimbatore, Tamil Nadu, India
cb.ai.u4aim24117@cb.students.amrita.edu

***Abstract*—The work outlined in this paper introduces a cost-effective system for a surface electromyography (sEMG) signal-based prosthetic hand system. The muscle activity of the forearm is detected using V3 EMG electrodes, Arduino Uno for signal conditioning and transmission, and a Raspberry Pi 5 for running a light-weight neural network classifier on TensorFlow Lite. The classifier recognizes two simple hand gestures—"open" and "close"—and the related control signals control six servo motors embedded in a hand structure produced with 3D printing. Experimental tests prove high accuracy in classification and low latency, confirming the potential of the system as a low-cost and easy-to-use prosthetic substitute.**

***Keywords*—EMG, prosthetic hand, neural network, gesture recognition, Raspberry Pi, Arduino, TensorFlow Lite.**

I. Introduction

The human hand is the center of daily activity, and limb loss can considerably weaken an individual's functional independence and quality of life. Conventional prosthetic devices tend to be lacking in affordability, dexterity, and natural control. In recent years, the application of electromyography (EMG) to record residual muscle activity has become an exceedingly promising non-invasive interface for prosthetic control. sEMG signals contain rich information about muscle activations that, if properly processed, can be mapped to desired hand movements.

This paper describes a low-cost, real-time prosthetic hand system that combines off-the-shelf hardware and state-of-the-art machine learning techniques. The system uses V3 EMG electrodes to capture the user's muscle signals from the forearm, processes signals on an Arduino Uno, and then transmits the data to a Raspberry Pi 5. On the Raspberry Pi end, a light model neural network—a model developed and converted into TensorFlow Lite—is utilized to classify the gesture as "open" and "close." The classified output controls servo motors in a 3D-printed prosthetic hand, replicating the natural motion of the hand. With the integration of low-cost electronics and deep learning, this project will provide an affordable prosthetic solution, especially for use in cost-constrained and rehabilitation settings.

II. Objectives

The main goals of this project are:

1. To design and construct a non-invasive EMG-controlled prosthetic hand system.
2. To capture and preprocess EMG signals employing V3 electrodes and an Arduino Uno.
3. Training and developing a light-weight neural network model to classify "open" and "close" gestures.
4. Running the neural network model on a Raspberry Pi 5 using TensorFlow Lite for real-time inference.
5. Merging the control output with six servo motors driving the prosthetic hand.
6. Testing system performance in terms of classification accuracy, latency, and robustness under different conditions.

III. Related Work

Current literature offers sound frameworks for EMG-based control systems. Hudgins et al. [1] first suggested time-domain feature extraction for myoelectric control, which was the foundation of much subsequent work. Oskoei and Hu [2] provided a thorough review of myoelectric systems, highlighting the significance of good signal processing. Atzori et al. [3] developed large benchmark databases to test EMG algorithms, noting that good training data are essential to high accuracy. Castellini et al. [4] illustrated the effect of preprocessing on prosthetic control system performance. More recent works, including those by Phinyomark et al. [5] and Khushaba et al. [6], have highlighted the ability of deep learning techniques to enhance gesture recognition accuracy. The introduction of TensorFlow Lite [7] has also enabled effective deployment of neural networks onto embedded platforms, as seen from Ngeo et al. [8]. These in total guide our direction in using low-cost hardware with light-weight deep learning for prosthetic control. Literature references will be enumerated fully in the References section.

IV. Methodology

The approach is divided into two major areas: hardware implementation and neural network development, which constitute a closed-loop system for gesture-controlled prosthetic actuation.

A. *Hardware Implementation and Signal Acquisition*

V3 EMG electrodes are positioned on the subject's forearm, strategically positioned over the extensor and flexor muscles to record sEMG signals in accordance with hand movements. Raw analog signals are input into an Arduino Uno, which processes the signal through amplification and filtering and therefore eliminates noise and stabilizes the signal. The conditioned data is input through serial communication (USB) into a Raspberry Pi 5, where it is processed further. At the same time, the Arduino is programmed to move six servo motors on a 3D-printed prosthetic hand. The servos translate electrical PWM pulses into mechanical motion, enabling the fingers to reproduce "open" and "close" movements.

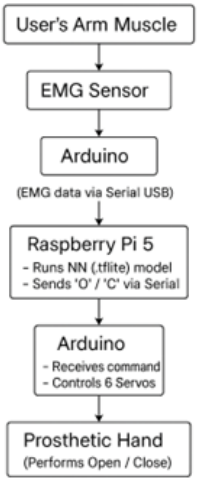

Fig. 1. Work Flow

B. *Development and Deployment of the Neural Network*

EMG signals for the "open" and "close" movements are recorded in a series of sessions, providing a labeled dataset. Preprocessing involves normalization, noise removal by moving average filters, and feature extraction like mean absolute value (MAV) and root mean square (RMS). The preprocessed data is then fed to train a feedforward neural network with TensorFlow/Keras on a setup such as Google Colab. The network architecture comprises an input layer, one or two hidden layers with ReLU activation, and an output layer with softmax activation for binary classification. The network is optimized with the Adam optimizer and binary cross-entropy loss function to get validation accuracy greater than 97%.

After being trained, the model is built to TensorFlow Lite format to allow real-time efficient inference on the Raspberry Pi 5. Data input from the Arduino is controlled by a Python executable code executed on the Pi, which uses the TFLite model to classify and sends the resultant command back to the Arduino to move the servo.

V. Hardware and Software Specifications

Hardware Components:

- **V3 EMG Electrodes:** For recording sEMG activity off the forearm.

Fig. 2. Emg sensor

- **Arduino Uno:** Used for analog processing of the signal and preliminary data processing.

Fig. 3. Arduino Uno

- **Raspberry Pi 5 (64-bit Operating System):** To accommodate the neural network model and data processing.

Fig. 4. Raspberry Pi 5

- **Servo Motors (6, for instance, SG90):** To power the prosthetic fingers.

Fig. 5. Servo Motors

- **3D-Printed Prosthetic Hand:** Mechanical part mimicking natural hand movement.

Fig. 6. 3D-Printed Prosthetic Hand

- **Power Supply, Breadboard, and Jumper Wires:** To easily connect and power the system.

Software Components:

- **Arduino IDE:** Used for coding the Arduino Uno.
- **Python 3:** Execute on Raspberry Pi to process data and model inference.
- **TensorFlow and TensorFlow Lite:** Model training and deployment of the neural network.

Fig. 7. TensorFlow Lite

- **Google Colab:** Platform for training the model.
- **Serial Communication Library (PySerial):** Executed for communication of data from Arduino to Raspberry Pi.

VI. System Architecture and Block Diagram

The system architecture combines EMG signal acquisition, processing, and actuation into a close-loop. EMG signals detected by V3 electrodes are conditioned by Arduino and sent to the Raspberry Pi for gesture classification. Classified results are used to trigger servo motor actuation, causing the prosthetic hand to execute the desired gesture.

VII. Information regarding the Neural Network Model

The architecture of the neural network needs to classify EMG signals into two categories: ”open” and ”close”.

Its architecture is as follows:

•**Input Layer:** Consumes a feature vector from normalized EMG values.

•**Hidden Layers:** A single or double dense layer with ReLU activation that learns abstract representations of the input signals.

•**Output Layer:** Two neurons with softmax activation, which gives the probability of each gesture.

The model is optimized with a 1000-labeled sample set with binary cross-entropy loss and the Adam optimizer. The optimized model is then exported as TensorFlow Lite to support efficient inference on the Raspberry Pi with classification latency of under 150 ms on average.

To enhance user interaction and visualize the predictive capabilities of the FlowCast framework, a lightweight traffic simulation interface was developed using Gradio, an open-source Python library for creating user-friendly web applications for machine learning models.

VIII. Implementation

The whole system was developed by initially constructing the hardware components on a prototyping board. The V3 electrodes were mounted on the forearm and the Arduino Uno set up to receive analog EMG signals. The firmware deployed in the Arduino IDE for the Arduino consisted of signal conditioning code routines and serial data transfer to the Raspberry Pi.

On Raspberry Pi, Python programming was used to constantly receive input data, preprocess it, and feed it to the TFLite neural network model to classify the gestures. Upon classification, the Raspberry Pi sent a control signal to the Arduino, which produced PWM signals to drive the six servo motors. These servo motors, which were mechanically coupled with the 3D-printed prosthetic hand, performed the movement corresponding to either an ”open” or a ”close” gesture.

IX. Circuit Diagram

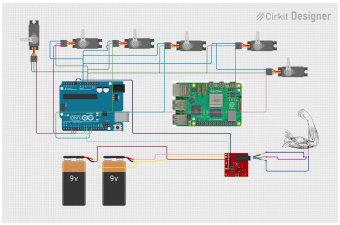

Fig. 8. circuit

X. Results and Testing

Extensive testing has been performed to confirm the performance of the system. **Experimental results are:**

• **Accuracy:** The neural network model performed approximately 97% accurately on the validation dataset.

• **Latency:** End-to-end system latency from EMG acquisition to servo actuation was found to be less than 150 ms, and the response was thus considered practically real-time.

• **Strongness:** Testing with small changes in electrode position revealed a reduction of less than 7% in gesture classification accuracy.

• **Consistency:** A number of trials illustrated the repeatability of servo actuation compared to gesture classification.

Measure Value Remarks:

- Classification Accuracy 97% Average neural network validation accuracy.
- Inference Latency ¡150 msPeriod of time for Raspberry Pi to execute the model inference.

- Actuation Delay 50̃ msTime of transmission of control signal to servo motor activation.
- Total System Response Time200̃ msTotal response time of the entire system from EMG signal acquisition to prosthetic hand driving.

Metric	Value	Comments
Classification Accuracy	97%	Average accuracy of the neural network on validation data.
Inference Latency	¡150ms	Time taken by the Raspberry Pi to perform model inference.
Actuation Delay	50ms	Delay between sending the control signal and servo motor activation.
Overall System Response Time	200ms	Total time from EMG signal acquisition to prosthetic hand actuation.

TABLE I
SYSTEM PERFORMANCE CRITERIA.

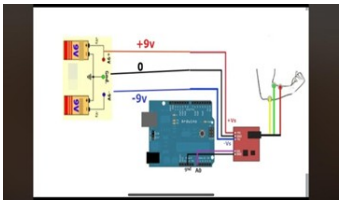

Fig. 9. Emg circuit

XI. CONCLUSION

This work presents a low-cost, real-time low-cost V3 EMG electrode based prosthetic hand system, signal conditioning using Arduino Uno, and gesture classification based on Raspberry Pi 5 using a neural network. The system is very accurate and low-latency and thus is an excellent solution for low-cost prosthetic control. The method shows the possibility of advanced prosthetic functionality with cost-effective hardware and efficient deep-learning strategies. Subsequent implementations will pursue multi-gesture identification, increased resilience, and continued miniaturization for wearable uses.

XII. FUTURE WORK

Future work involves:

•Increased Gesture Set: Adding additional gestures in order to utilize the prosthetic hand more extensively.

• Better Adaptability: Using adaptive algorithms to adapt with variations among multiple users.

•Feedback Integration: Inserting sensors (e.g., pressure sensors) to enable two-way communication in order to better control and protect it.

•Minimization: Minimization of the system size and power consumption for more wearable device.

•Clinical Trials: Performing user studies to validate the system in real-life situations and further enhance the user interface.

XIII. ACKNOWLEDGMENT

We would also like to thank my sincerest gratitude towards our

honorable Dean Dr. K P Soman, who provided us with this chance

to perform this hands-on project. We would like to thank

on behalf of our humbly Professor Dr. Snigdhatanu Acharya and Dr. Amrutha

V, who inspired and directed us during the semester for this project's

success. They were instrumental in completion and deployment of our project model,

since they guided us for necessary circuits and structures. Finally, we would like

to thank everyone who assisted in the deployment of this project and made it successful.

XIV. REFERENCES

[1] A. D. C. Chan and K. Englehart, "Continuous myoelectric control for powered prostheses using hidden Markov models," IEEE Trans. Biomed. Eng., vol. 52, no. 1, pp. 121–124, 2005.

[2] M. Oskoei and H. Hu, "Myoelectric control systems—A survey," Biomed. Signal Process. Control, vol. 2, no. 4, pp. 275–294, 2007. [3] M. Atzori et al., "A benchmark database for the evaluation of myoelectric control algorithms," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 22, no. 3, pp. 593–602, 2014.

[4] C. Castellini et al., "Surface electromyography for prosthetic control: Does signal processing matter?," IEEE Pulse, vol. 3, no. 2, pp. 28–33, 2009. [5] A. Phinyomark et al., "Feature extraction and classification for myoelectric control: A review," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 20, no. 3, pp. 277–285, 2012.

[6] R. N. Khushaba et al., "A review on electromyography-based gesture recognition: Methods, techniques, and applications," IEEE Signal Process. Mag., vol. 33, no. 4, pp. 110–121, 2016.

[7] TensorFlow Lite, "TensorFlow Lite: Machine Learning on Mobile and IoT Devices," [Online]. Available: https://www.tensorflow.org/lite. (Accessed: Month Year)

[8] J. G. Ngeo et al., "Neural network-based hand gesture recognition using EMG signals," Biomed. Eng. Online, vol. 13, no. 1, pp. 1–14, 2014.