ExoPower

--Your Muscles Your Partner

Innovating for a sustainable future

Presenter Name

Content

Introduction

ExoPower Exoskeleton Arm:

- Driven by motor
- EMG sensoring
- Mechanical structure fits to human arm movements
- Fully automatic code control

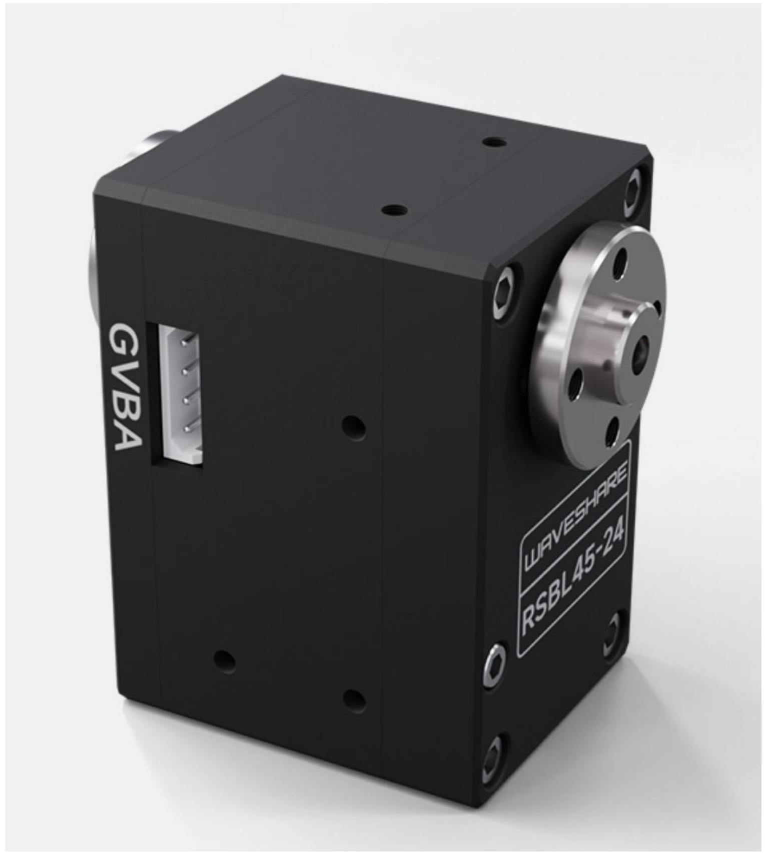
ExoPower

Components Introduction

Component Introduction--Motor

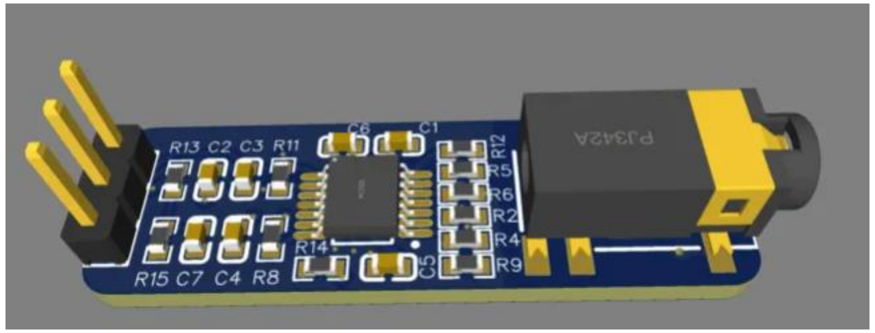
DYNAMIXEL XM430-W210-T

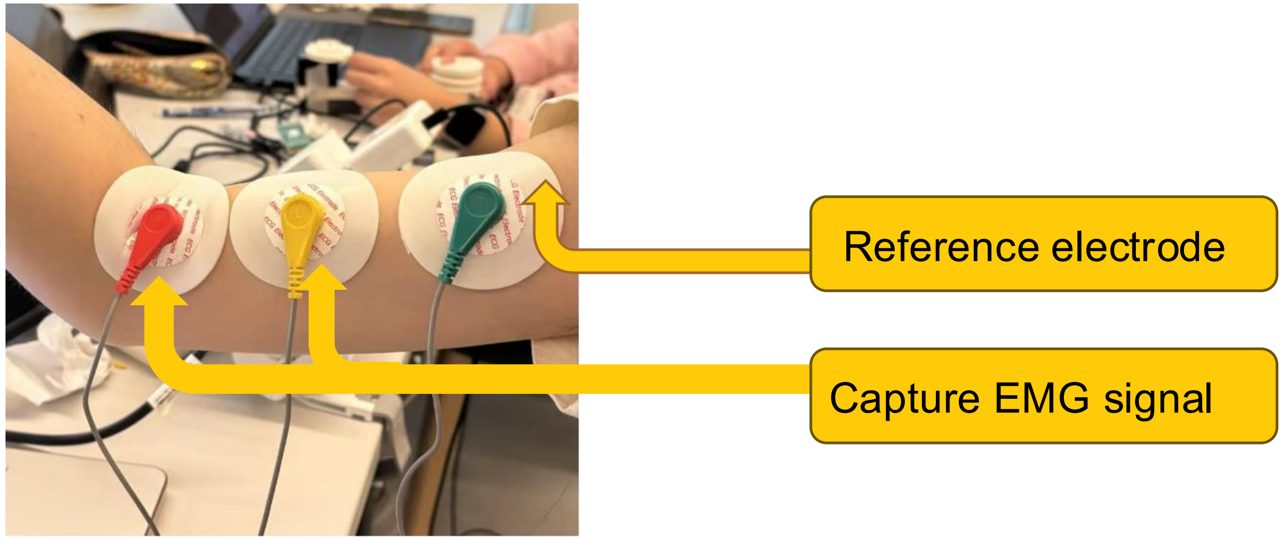
Performance	XM430-W210-T	Relevance to Exoskeleton Function
Peak Torque	3.0 N·m (@12V)	Enables the exoskeleton to assist in lifting objects, supporting practical weight loads.
Control Modes	6 modes (position, speed, current, etc.)	Allows precise adjustment to match user's needs via EMG signals, ensuring smooth a nd adaptive operation.
Encoder Resolution	12-bit, contactless encoder (0.088° × 4096)	Ensures fine movement control and accur acy in exoskeleton actions.
Communication	TTL/RS-485, multi-node support	Supports coordination between multiple motors for complex multi-degree-of-freedom movements.
Weight & Size	82g; 28.5×46.5×34 mm	Lightweight and compact, ideal for weara ble exoskeletons, enhancing comfort and mobility.
Structural Features	Hollow design, flexible wiring, durable casing	Provides structural stability and ease of wi ring, enhancing both durability and comfo rt.



Component Introduction-- Sensor

Electromyography (EMG) is a technique for assessing and recording the electrical activity produced by skeletal muscles

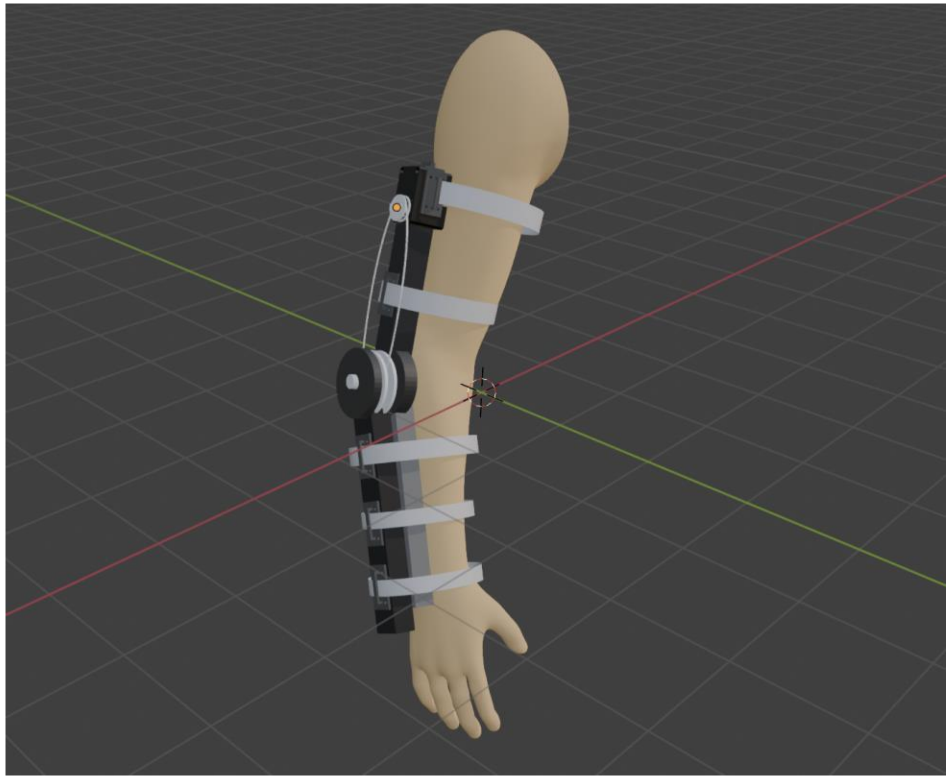




Prototype

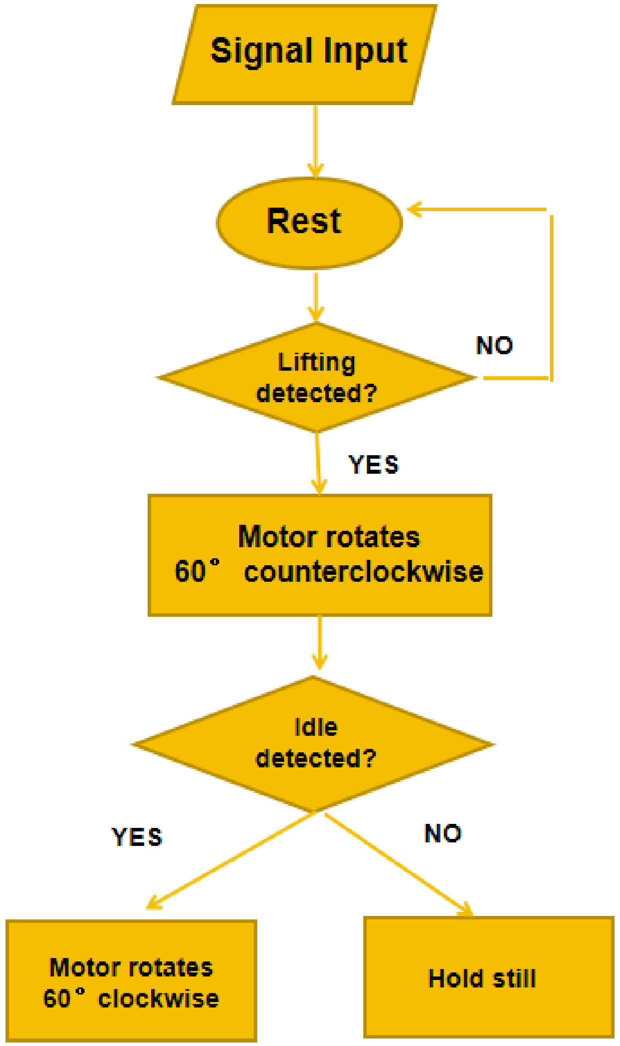
Exoskeleton Arm Model Description

- Mechanical Design: Pulley System
  - One on shoulder, another one on elbow
  - Using Nylon String to connect
  - 3D-printed frame



Structure Designing--Mechanical working principle

- Driven by EMG Signal:
  - Rest Signal: keep the arm resting
  - Lifting Signal: Trigger motor doing lifting motion



System Development Progress

Code Architecture

Data Collection Module: Reads EMG signals via Arduino and labels them for training.

Model Module: Trains and loads a classifier to predict muscle activity.

Real-Time Module: Continuously processes incoming signals and runs prediction.

Motor Control Module: Sends serial commands to control the Dynamixel motor based on predictions.

GUI Module: Visualizes EMG signals and prediction results using PyQtGraph.

Data Collection

Collect real-time EMG signals using an Arduino board and EMG sensor.

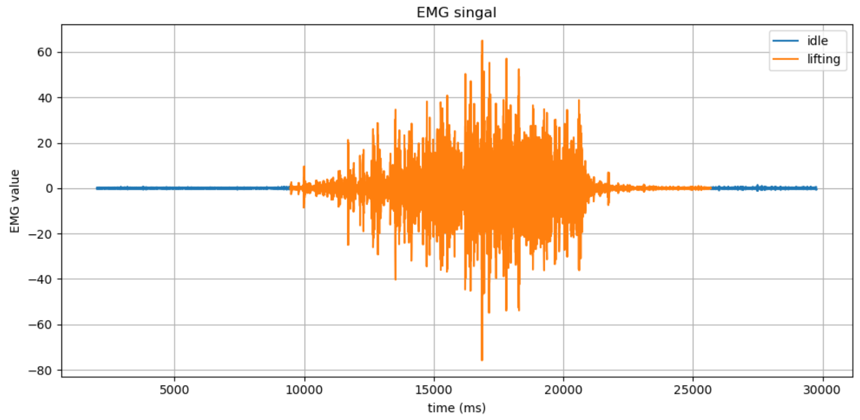
Raw signals are transmitted via serial port to a Python script that records, labels, and saves them as CSV files.

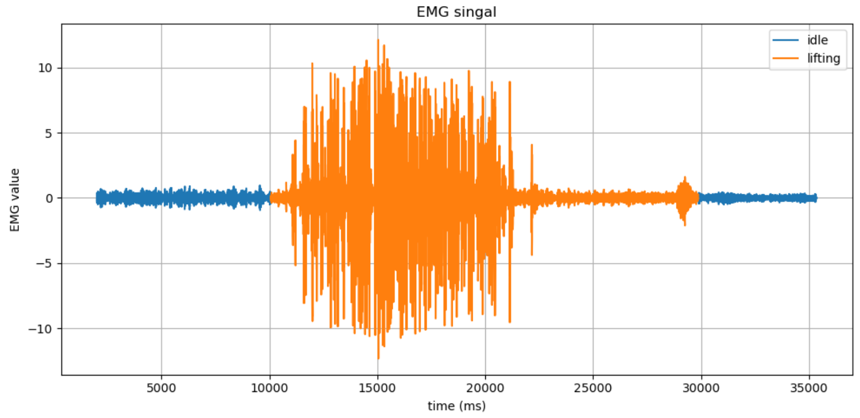
Label switching: Use spacebar to toggle action labels (idle / lifting)

Live tagging: Each sample includes timestamp, EMG value, and label

Visualization: A plot is automatically generated and saved after recording

Saved files: CSV for raw data, PNG for visualization

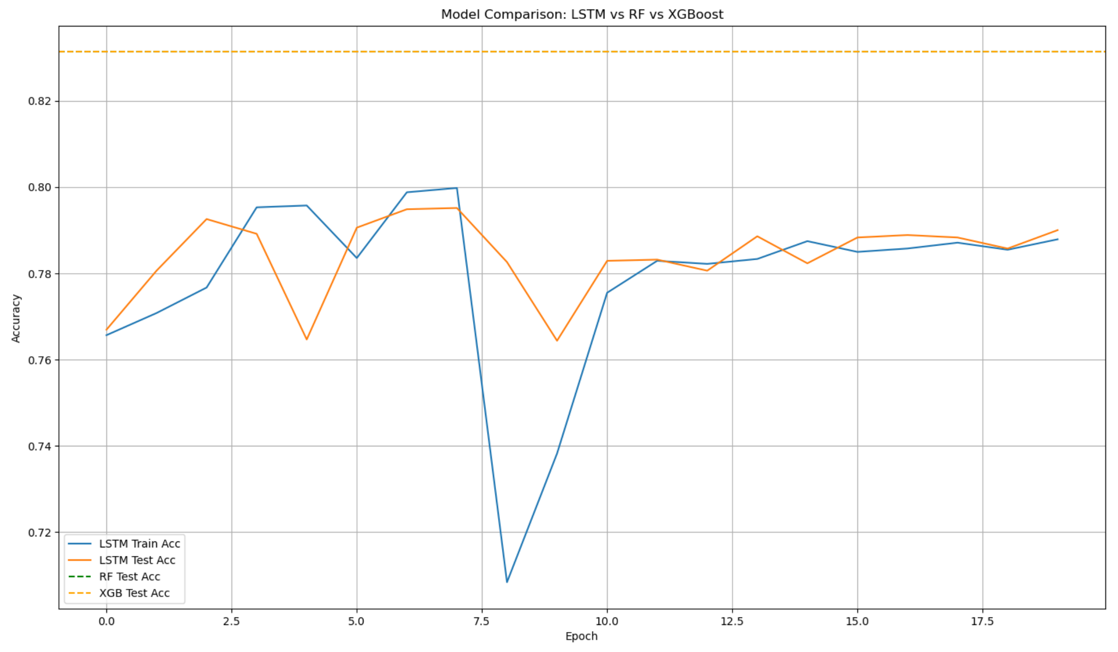




Model Training

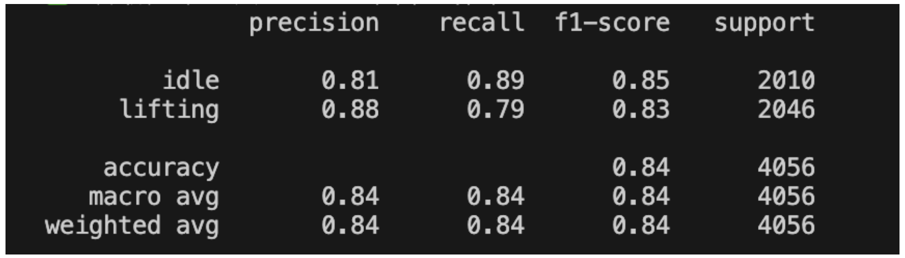
Final Choice: Random Forest (RF) for real-time classification

- High accuracy, close to XGBoost
- Stable performance, no significant fluctuations
- Fast inference, suitable for real-time use
- Simple implementation, easy to migrate and debug

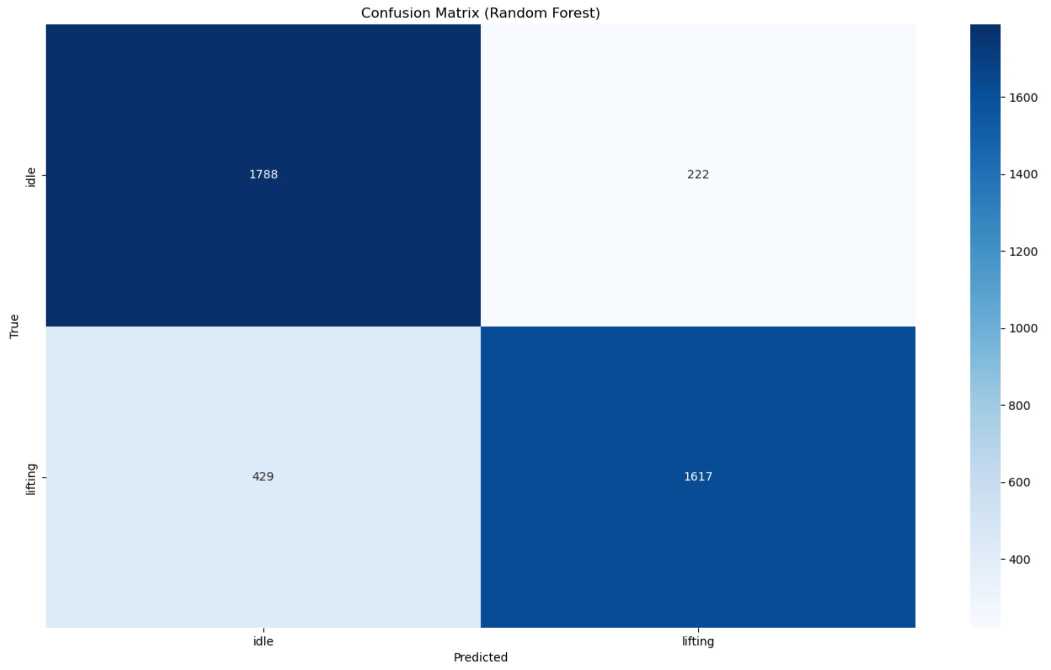


Model Training

- EMG signals are windowed (size = 100)
- 7 features extracted: MAV, RMS, WL, ZC, SSC, Skewness, Kurtosis
- Trained using RandomForestClassifier(n_estimators=200)
- Test accuracy reached about 83.3%



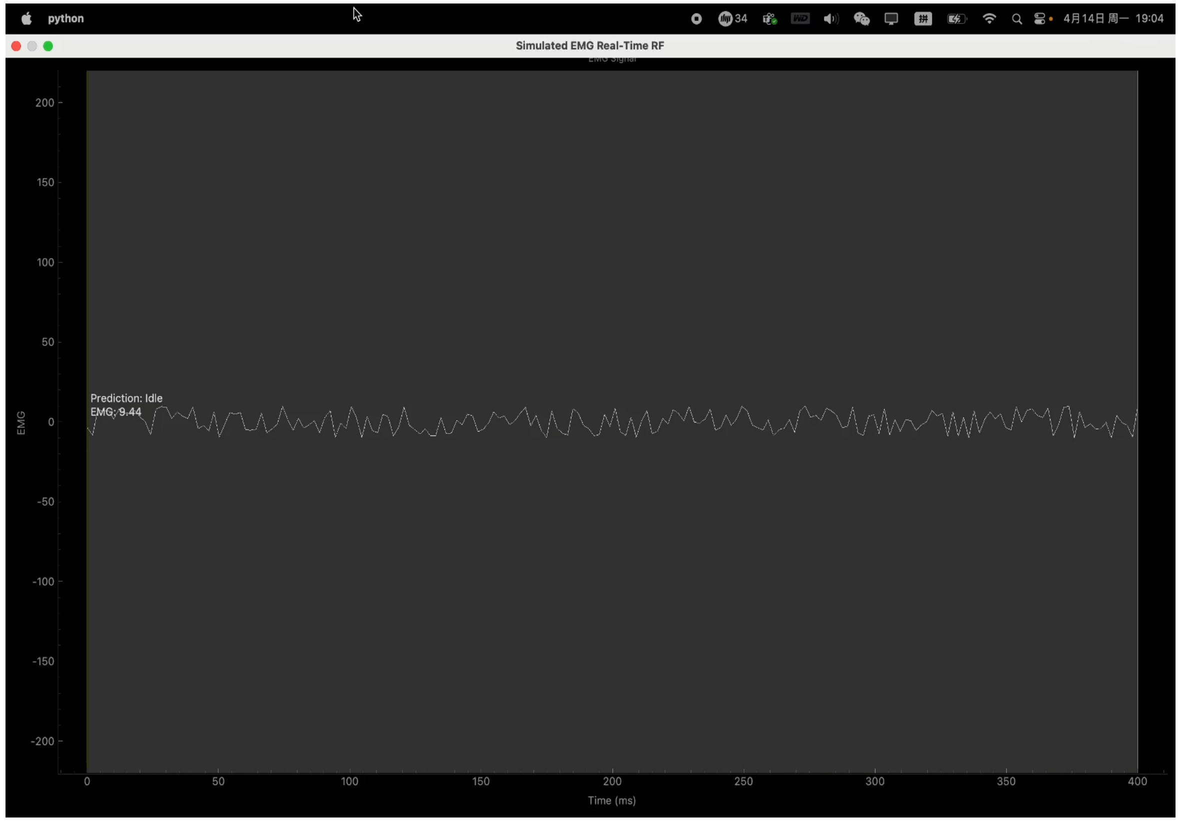
```
              precision    recall  f1-score   support

        idle       0.81      0.89      0.85      2010
     lifting       0.88      0.79      0.83      2046

    accuracy                           0.84      4056
   macro avg       0.84      0.84      0.84      4056
weighted avg       0.84      0.84      0.84      4056
```



Evaluation: Confusion matrix shows clear separation between idle and lifting

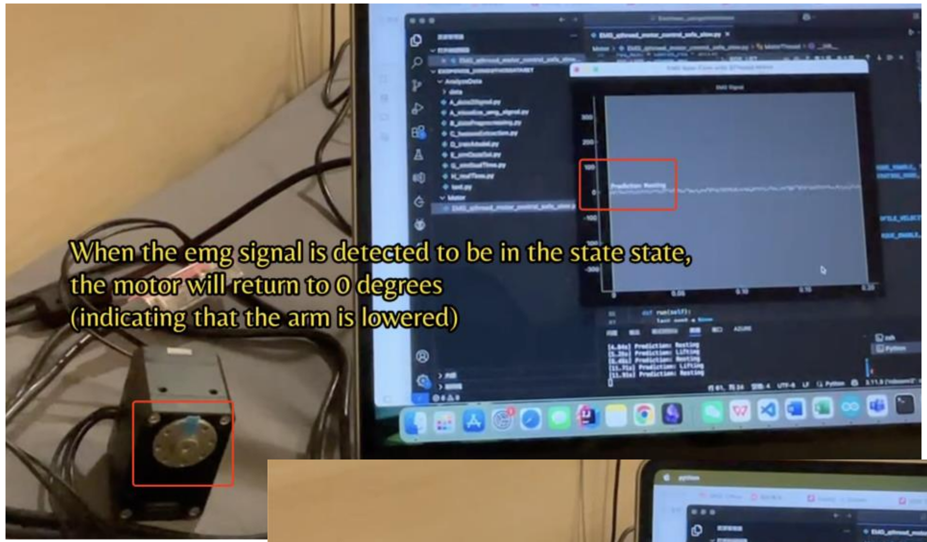
Deployment of Real-Time EMG Recognition

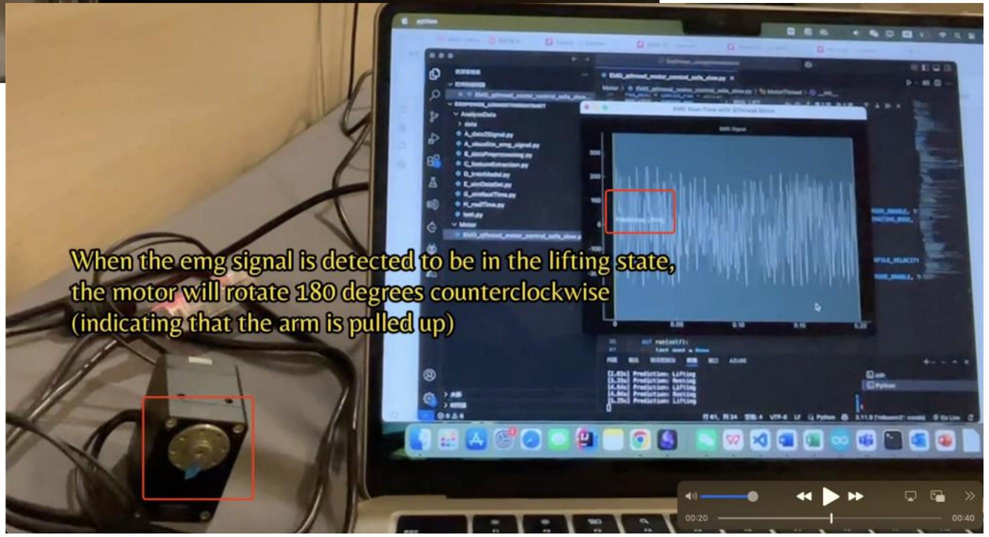
- Connected real EMG sensor via serial port (Myoware)
- Real-time EMG signal received and plotted (500Hz sampling)
- Feature extracted every 100 points, prediction made using RF model
- Background color changes with predicted state (Idle / Lifting)
- Sends control commands to motor (e.g., MOTOR_FORWARD)



Using EMG Prediction to Control the Motor

- When **Lifting** is predicted continuously → motor lifts the robotic arm
- When **Idle** is detected → motor rotates clockwise to lower the arm
- Serial commands are sent: MOTOR_FORWARD / MOTOR_BACKWARD
- A state buffer ensures stability by requiring repeated predictions
- This achieves a closed loop:
  EMG signal → action classification → motor control → arm movement


When the emg signal is detected to be in the state state,
the motor will return to 0 degrees
(indicating that the arm is lowered)


When the emg signal is detected to be in the lifting state,
the motor will rotate 180 degrees counterclockwise
(indicating that the arm is pulled up)

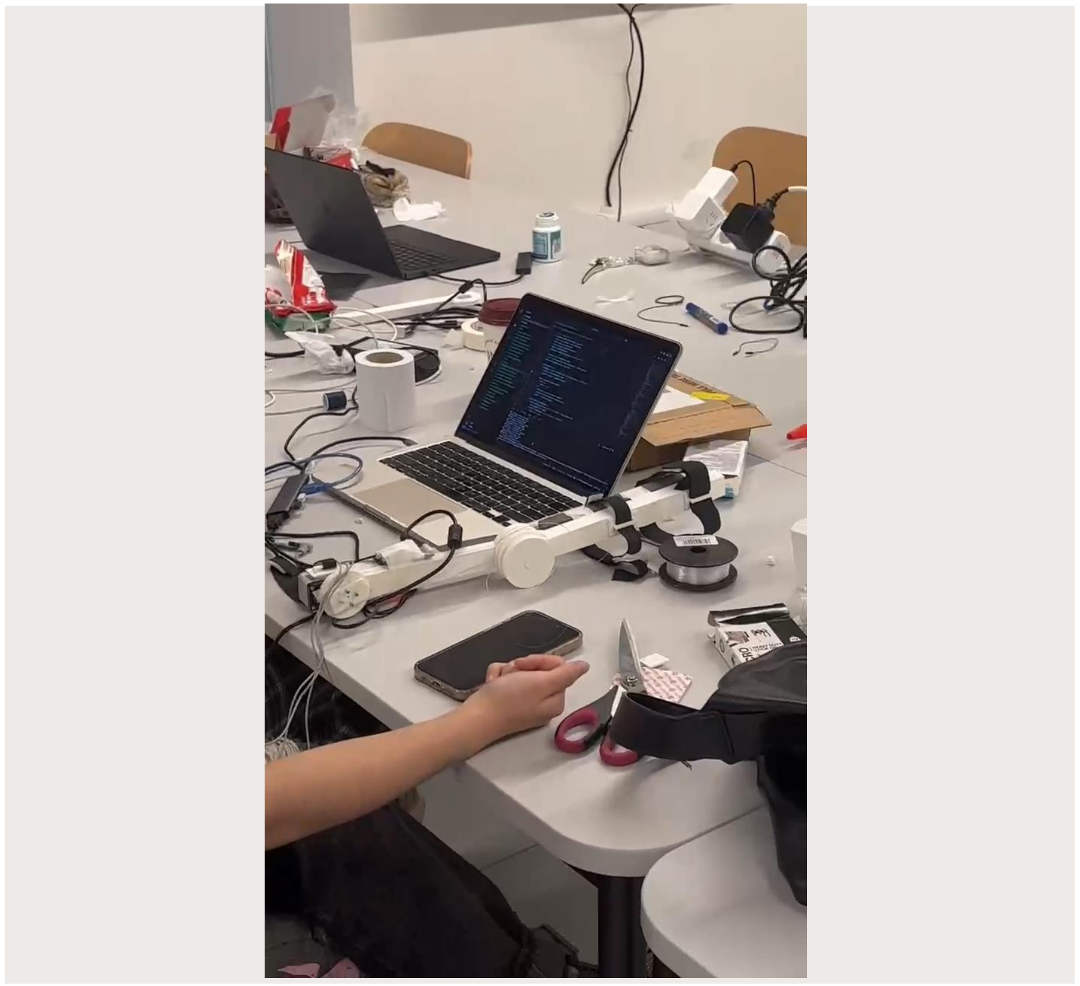
EMG Signal Integration with Motor Control

Multithreaded architecture ensures smooth real-time performance:

- SerialReaderThread: continuously reads EMG input
- MotorThread: controls actuator based on prediction
- Main thread: runs prediction + GUI rendering

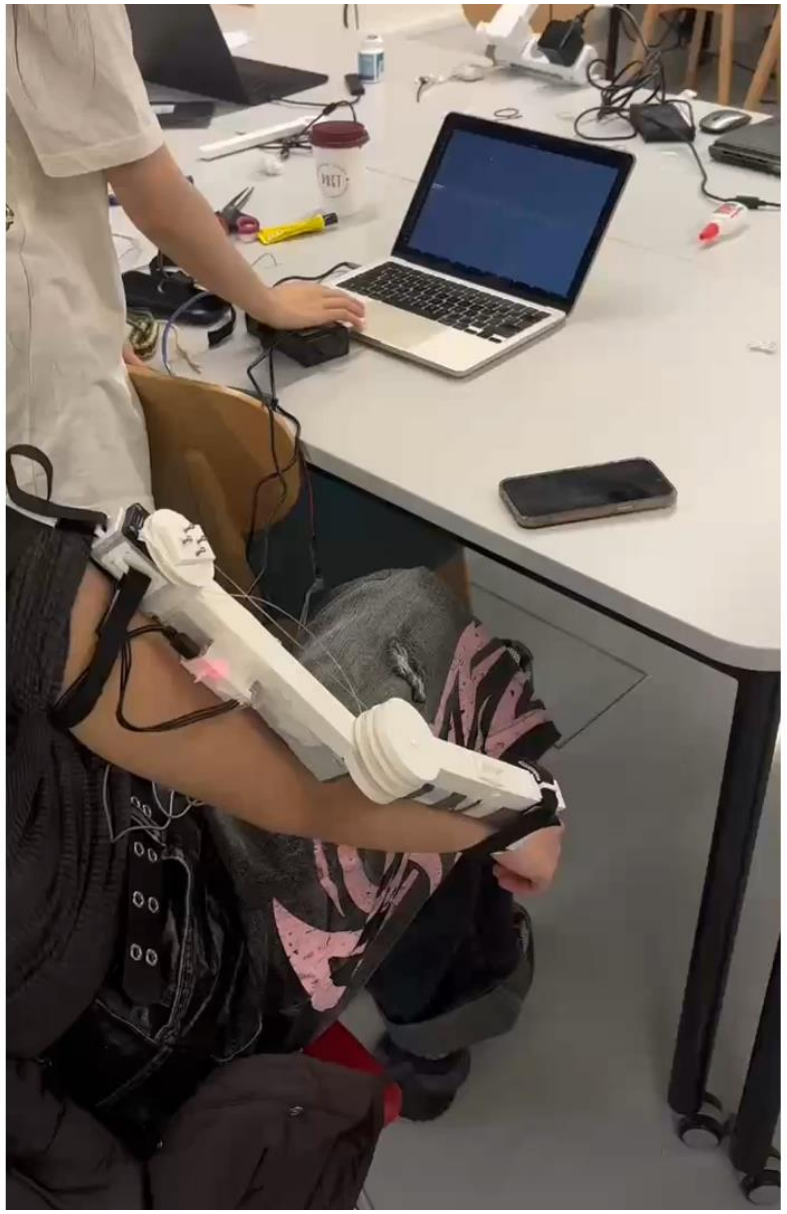
Real-time classification using Random Forest (joblib)

State buffer ensures action stability and prevents false triggers



Demo Showcase

Live Demo



Future Goals

Short-term Goals & Long-term Vision

Short-term Objectives

01

Expansion of working modalities

Daily movement assistance mode and sports exercise assistance mode
combined with the IMU sensor: sense the movement status

02

Incorporating adaptive control algorithms

Add wifi module, use Bluetooth/WiFi control: Using Azure make Visualization Dashboard

03

Wireless control and data transmission

Incorporate adaptive control algorithm; Enables closed-loop control

04

Enabling Personalized Adaptation

Adapting with parameters adjusted according to the muscle strength or physiological characteristics of different users

Thank you