EMG-Controlled Exosuit Setup Guide

1 Project Overview

This guide provides complete setup instructions for the EMG-controlled exosuit system, which uses Myo armband data to control an exosuit via machine learning models.

2 Prerequisites

- Python 3.7+
- Android Studio
- Myo Armband
- Raspberry Pi (for deployment) or WSL (for development)

3 Step 1: Repository Setup

1. Clone the main repository:

```
    git clone <https://gitlab.aibe.fau.de/airob/theses/research-lab-dawood-mughal>
    cd <research-lab-dawood-mughal>

```

2. Install required Python dependencies:

```
    pip install -r requirements.txt

```

4 Step 2: Training Server Setup

4.1 Training Server Configuration

1. Navigate to the TrainerUDPServer folder:

```
    cd TrainerUDPServer

```

2. The file `wrist_exo_model_trainer.py` serves as the training server with UDP communication instead of traditional HTTP calls.

3. Key features of the training server:
   - Receives EMG recorded CSV files via UDP from Android
   - Trains machine learning models based on configurable feature sets
   - Implements `find_best_model()` function for model selection
   - Allows customization of feature sets within this function

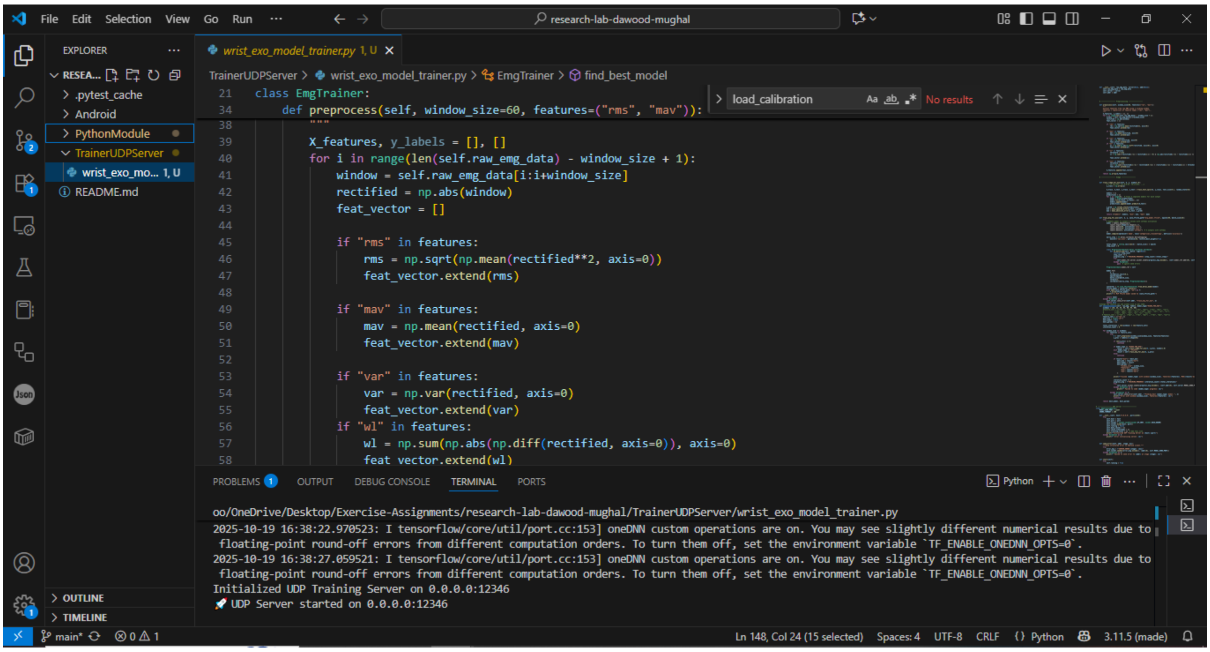
4.2 Running the Training Server

1. Run the training server directly (not in WSL):

```
python wrist_exo_model_trainer.py

```

2. Configure firewall rules to allow ports: 3350, 3352, 3358, 12346, 12347
3. Note the IP address where the training server is running
4. Update the Android module with this IP address in `UdpMotorController.getTrainingServerIp()`



```
class EmgTrainer:
    def preprocess(self, window_size=60, features=("rms", "mav")):
        """
        X_features, y_labels = [], []
        for i in range(len(self.raw_emg_data) - window_size + 1):
            window = self.raw_emg_data[i:i+window_size]
            rectified = np.abs(window)
            feat_vector = []

            if "rms" in features:
                rms = np.sqrt(np.mean(rectified**2, axis=0))
                feat_vector.extend(rms)

            if "mav" in features:
                mav = np.mean(rectified, axis=0)
                feat_vector.extend(mav)

            if "var" in features:
                var = np.var(rectified, axis=0)
                feat_vector.extend(var)
            if "wl" in features:
                wl = np.sum(np.abs(np.diff(rectified, axis=0)), axis=0)
                feat_vector.extend(wl)
```

```
oo/OneDrive/Desktop/Exercise-Assignments/research-lab-dawood-mughal/TrainerUDPServer/wrist_exo_model_trainer.py
2025-10-19 16:38:22.970523: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to
 floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-10-19 16:38:27.059521: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to
 floating-point round-off errors from different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
Initialized UDP Training Server on 0.0.0.0:12346
🚀 UDP Server started on 0.0.0.0:12346
```

Figure 1: Training Server Configuration

5 Step 3: Motor Controller Setup

5.1 CANdle and Motor Connections

1. Connect motors to each other in the designated sequence
2. Connect motors to the CANdle device
3. Plug in the CANdle device until the motor displays a blinking light indicating connection

5.2 Option 1: Raspberry Pi Deployment

1. Navigate to the controller directory:

```
cd pythonModule/controller

```

2. Modify configuration files:
   - Set `TEST_MODE = False` in `exo_controller_*.py`
   - Set `TEST_MODE = False` in `main.py`
   - Set `TEST_MODE = False` in `utils.py`
3. Run the main controller:

```
python main.py

```

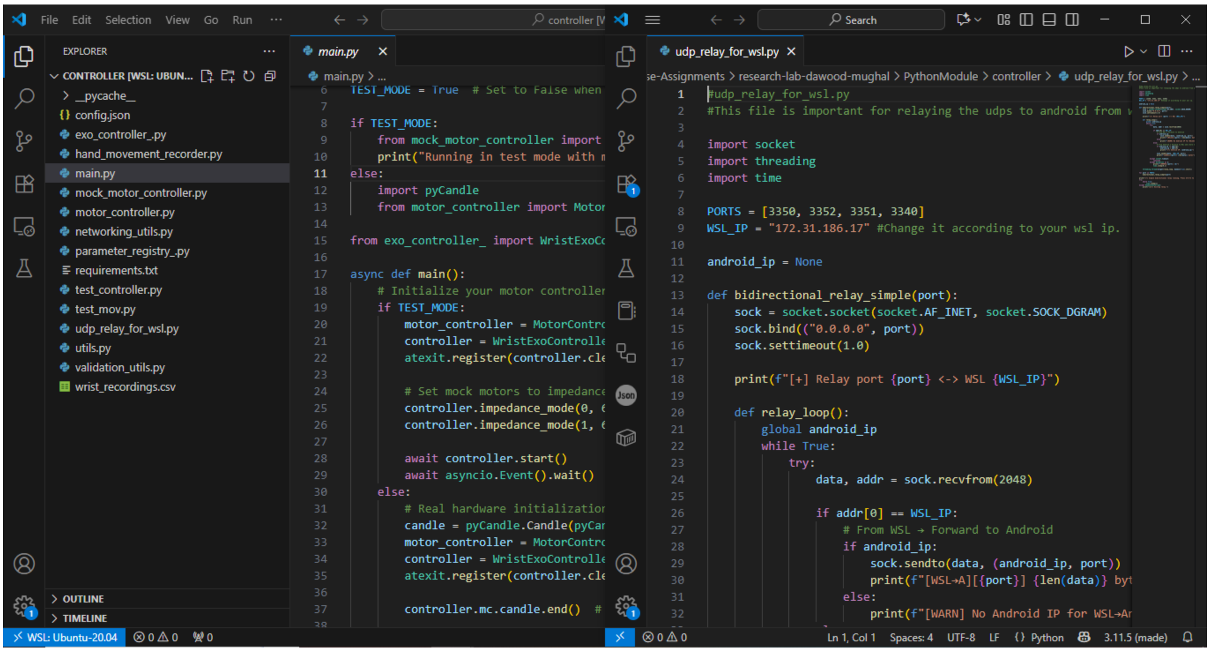
5.3 Option 2: WSL Development Setup

1. **Open WSL in VS Code:**
   - Open VS Code
   - Press `Ctrl + Shift + P` and type ”Remote-WSL: New Window”
   - Select ”Remote-WSL: New Window using WSL”
   - In the new WSL window, open the path: `/mnt/c/path/to/PythonModule/controller`
2. Run `main.py` within WSL
3. Additionally run the UDP relay script outside WSL by making a new window as shown in Figure 2:

```
    python udp_relay_for_wsl.py

```

4. Update `WSL_IP` in `udp_relay_for_wsl.py` with your WSL IP address
5. Note the system IP address (not WSL IP) for Android configuration



```
TEST_MODE = True  # Set to False when

if TEST_MODE:
    from mock_motor_controller import
    print("Running in test mode with m
else:
    import pyCandle
    from motor_controller import Motor

from exo_controller_ import WristExoCc

async def main():
    # Initialize your motor controller
    if TEST_MODE:
        motor_controller = MotorContrc
        controller = WristExoControlle
        atexit.register(controller.cle

        # Set mock motors to impedance
        controller.impedance_mode(0, €
        controller.impedance_mode(1, €

        await controller.start()
        await asyncio.Event().wait()
    else:
        # Real hardware initializatior
        candle = pyCandle.Candle(pyCar
        motor_controller = MotorContrc
        controller = WristExoControlle
        atexit.register(controller.cle

        controller.mc.candle.end()  #
```

```
#udp_relay_for_wsl.py
#This file is important for relaying the udps to android from w

import socket
import threading
import time

PORTS = [3350, 3352, 3351, 3340]
WSL_IP = "172.31.186.17" #Change it according to your wsl ip.

android_ip = None

def bidirectional_relay_simple(port):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind(("0.0.0.0", port))
    sock.settimeout(1.0)

    print(f"[+] Relay port {port} <-> WSL {WSL_IP}")

    def relay_loop():
        global android_ip
        while True:
            try:
                data, addr = sock.recvfrom(2048)

                if addr[0] == WSL_IP:
                    # From WSL → Forward to Android
                    if android_ip:
                        sock.sendto(data, (android_ip, port))
                        print(f"[WSL→A][{port}] {len(data)} byt
                    else:
                        print(f"[WARN] No Android IP for WSL→Ar
```

Figure 2: Motor Controller Setup

6 Step 4: Android Application Setup

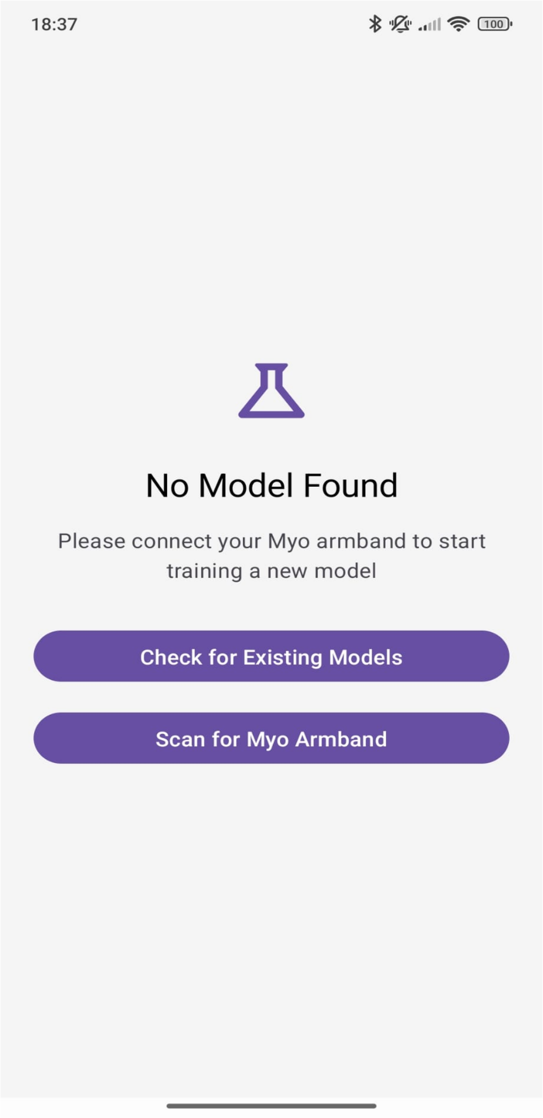
6.1 Application Configuration

1. Open the project in Android Studio
2. Navigate to `UdpMotorController` in the utility folder
3. Update IP addresses in two locations:
   - `getRaspiServerIp()` - Motor controller IP
   - `getTrainingServerIp()` - Training server IP
4. Build and install the application on Android device

7 Step 5: System Operation

7.1 Initial Setup and Training

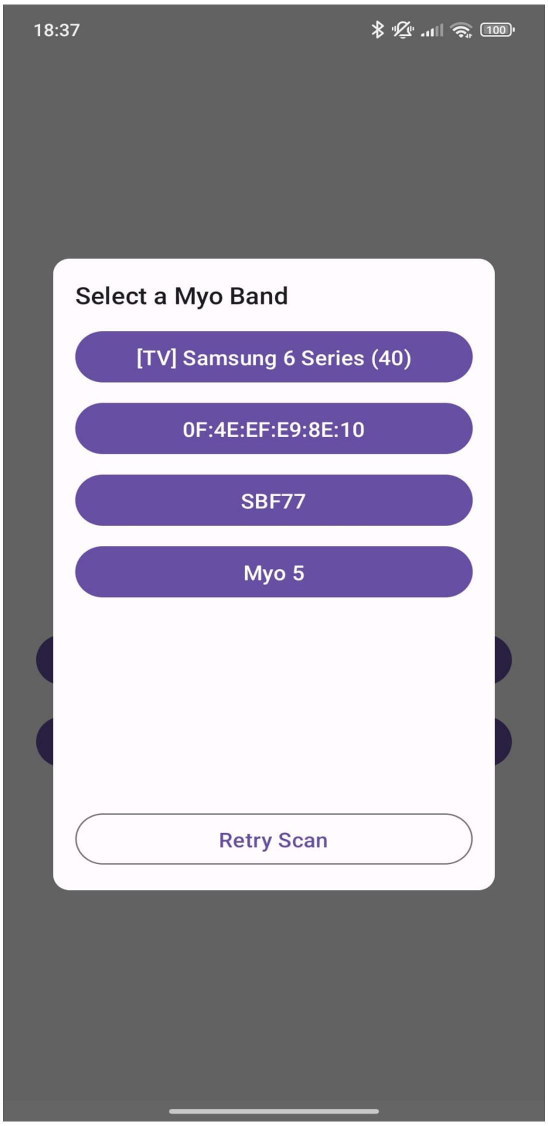
1. Launch the Android application
2. Accept required permissions
3. Scan for Myo Armband
4. Connect Myo armband and wear it properly
5. Click ”Start Training Session” button



No Model Found

Please connect your Myo armband to start training a new model

Figure 3: First Screen

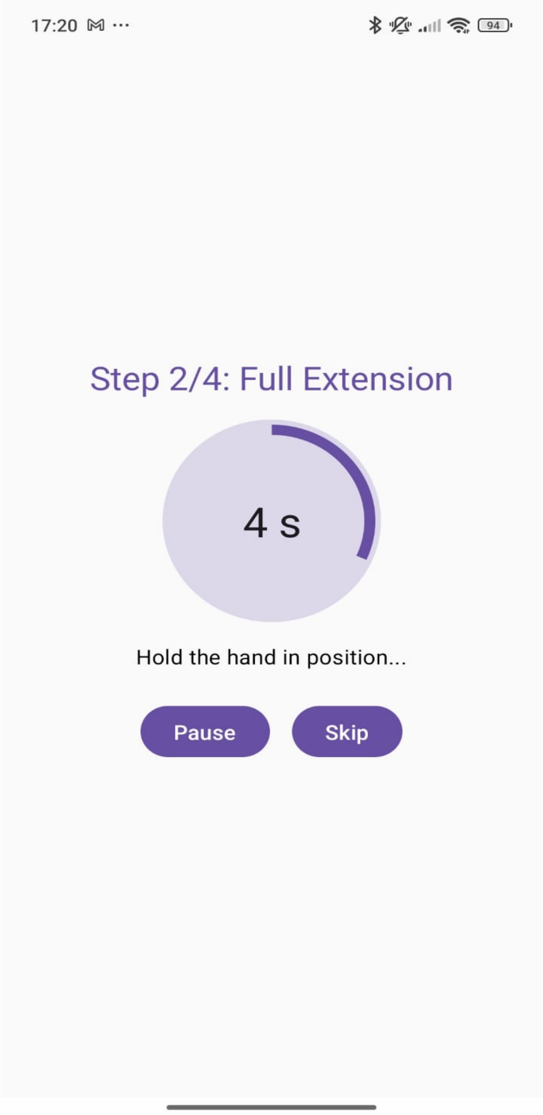


Select a Myo Band

Figure 4: Connect Myo Armband

7.2 Data Recording Process

1. Follow guided recording screen for EMG data collection
2. Perform movements as shown (flexion, extension, etc.)
3. After recording, choose Save data and train model:
   - Save data and train model
   - Save data only
   - Discard data
4. Select training algorithm (Ridge Regression or MLP)
5. Monitor training progress on screen

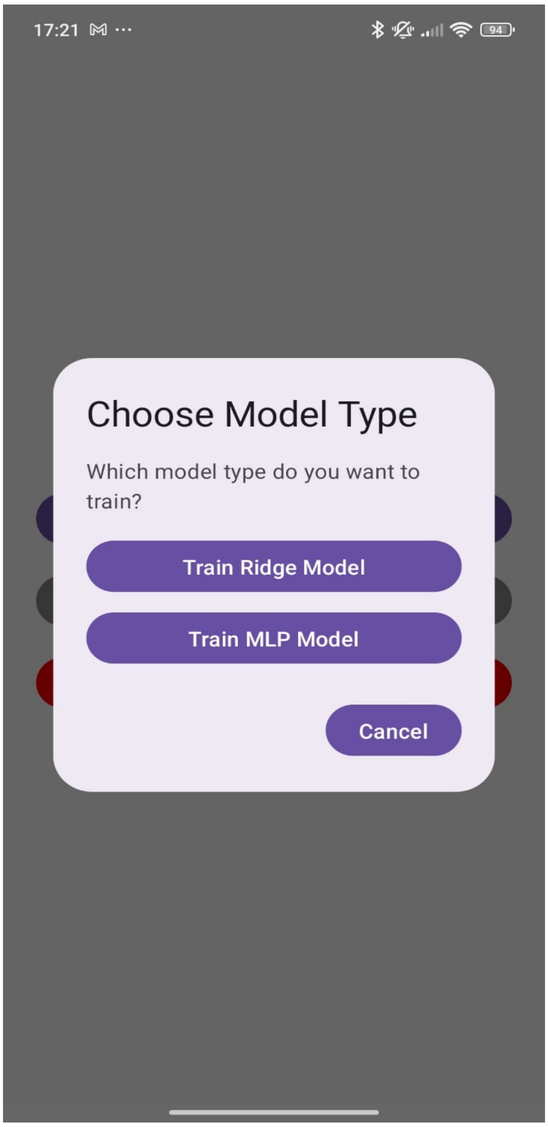


Step 2/4: Full Extension

4 s

Hold the hand in position...

Figure 5: EMG Data Recording step



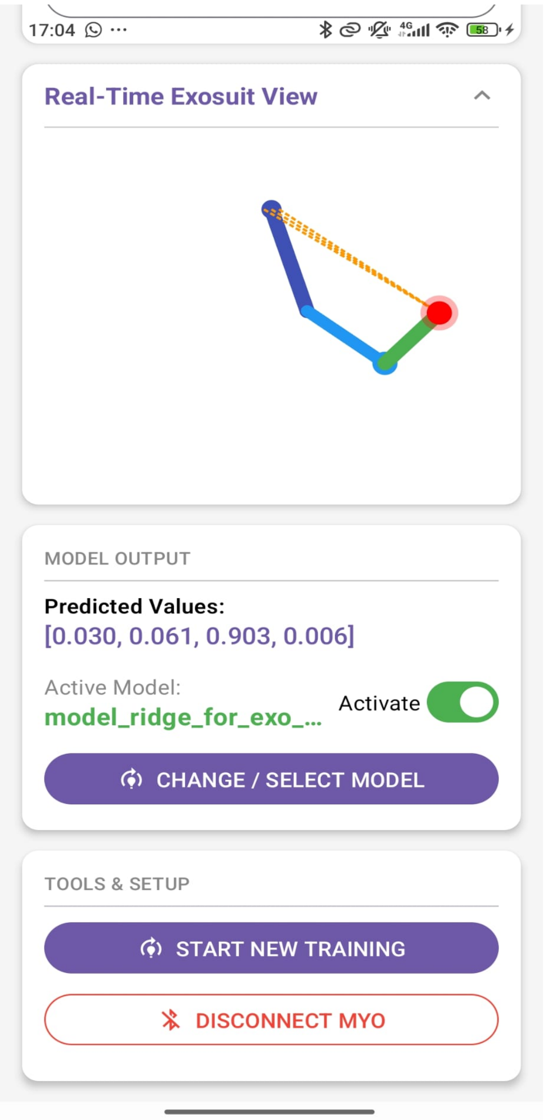
Choose Model Type

Which model type do you want to train?

Figure 6: Model Selection

7.3 Select Trained Model and Activate

1. Return to home screen after completion
2. Use "Change/Select Model" button to choose trained model
3. Enable "Active Model" switch



Predicted Values:
[0.030, 0.061, 0.903, 0.006]

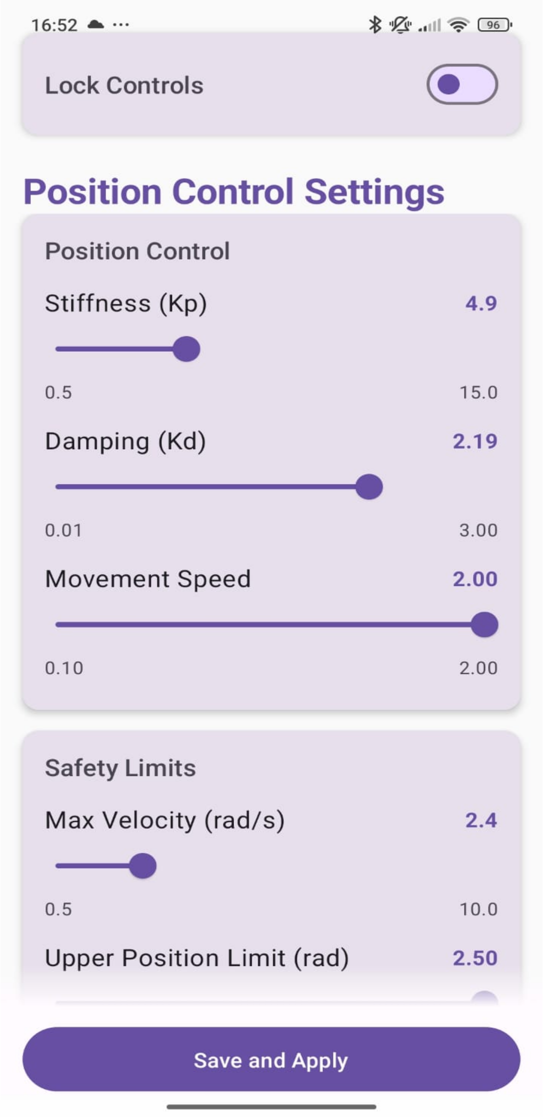
Active Model:
model_ridge_for_exo_...

Figure 7: Model Selection Interface

7.4 Exosuit Activation and Control

1. Click "Motor Settings" on main screen
2. Adjust motor parameters as needed
3. Save and apply settings (sends initial configuration)
4. Monitor connection status:
   - Initially: Disconnected
   - After settings applied: Status becomes READY TO START

5. Press "Start System" button to begin real-time control
6. Use activate model switch to pause sending predictions.
7. Use disconnect button to terminate connection



16:52

Lock Controls

Position Control Settings

Position Control

Stiffness (Kp) 4.9

0.5 15.0

Damping (Kd) 2.19

0.01 3.00

Movement Speed 2.00

0.10 2.00

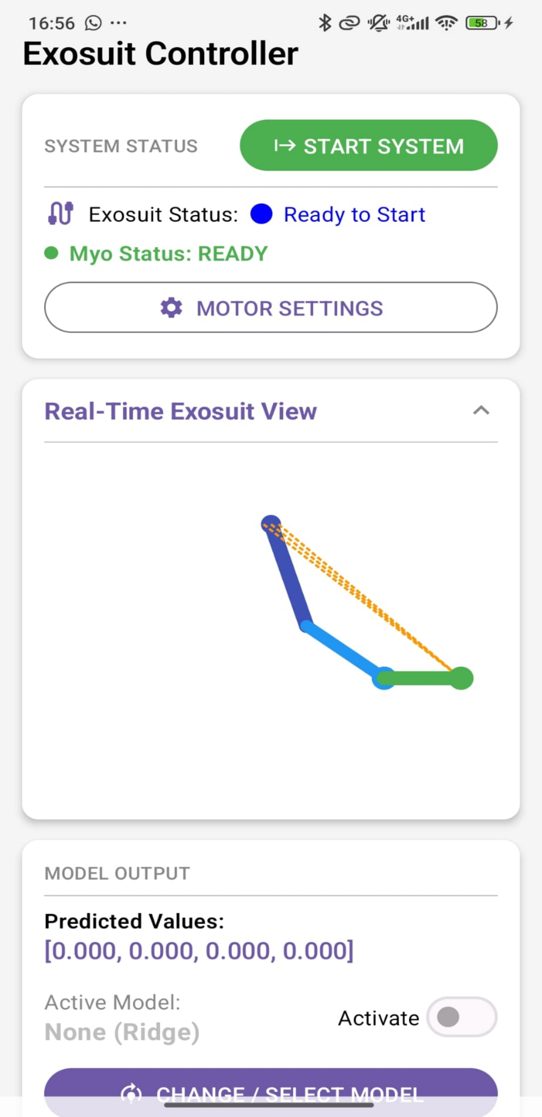
Safety Limits

Max Velocity (rad/s) 2.4

0.5 10.0

Upper Position Limit (rad) 2.50

Figure 8: Motor Settings



16:56

Exosuit Controller

SYSTEM STATUS

Exosuit Status: ● Ready to Start

● Myo Status: READY

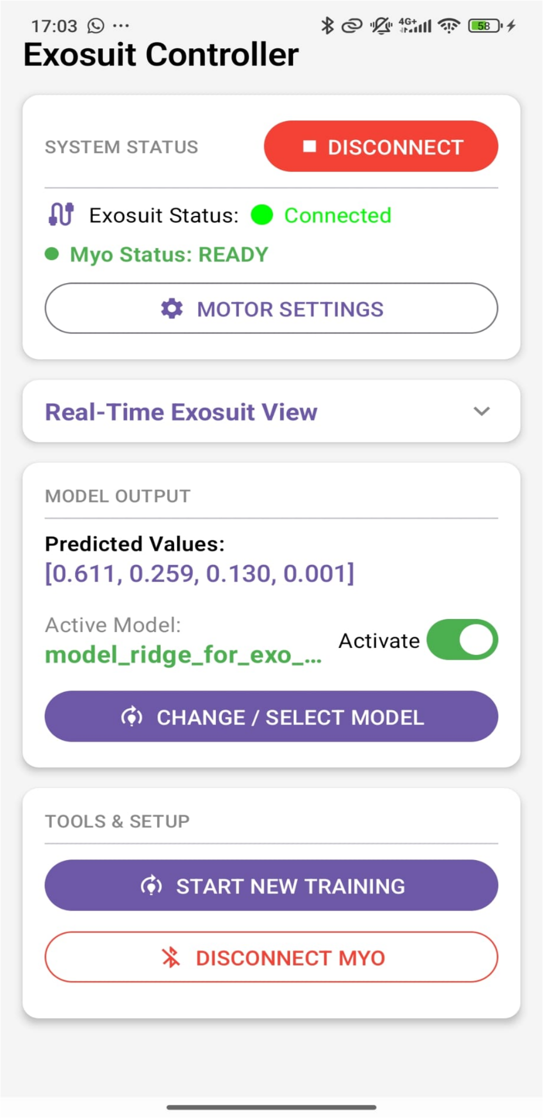
MODEL OUTPUT

Predicted Values:
[0.000, 0.000, 0.000, 0.000]

Active Model:
None (Ridge)

Activate

Figure 9: Ready To Start System



17:03

Exosuit Controller

SYSTEM STATUS

Exosuit Status: ● Connected

● Myo Status: READY

Real-Time Exosuit View

MODEL OUTPUT

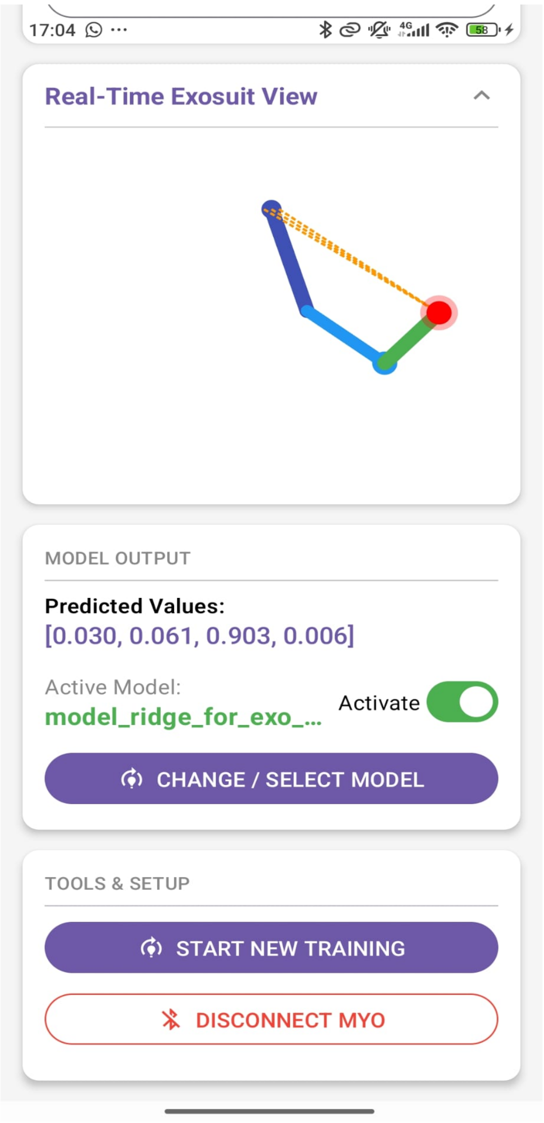
Predicted Values:
[0.611, 0.259, 0.130, 0.001]

Active Model:
model_ridge_for_exo_...

Activate

TOOLS & SETUP

Figure 10: Main Control Interface



17:04

Real-Time Exosuit View

MODEL OUTPUT

Predicted Values:
[0.030, 0.061, 0.903, 0.006]

Active Model:
model_ridge_for_exo_...

Activate

TOOLS & SETUP

Figure 11: Real-Time View

8 Troubleshooting

8.1 Common Issues

- **Firewall blocking ports**: Ensure ports 3350, 3352, 3358, 12346, 12347 are open. If there is a problem with UDP communication even after adding inbound rule try disabling Firewall for all.
- **Connection timeout**: Verify IP addresses in Android configuration
- **Training failures**: Check feature set configuration in `find_best_model()`
- **Motor not responding**: Confirm TEST_MODE is set to False in controller files

8.2 Connection Status Indicators

- **Red**: Disconnected
- **Yellow**: Connecting
- **Green**: Connected and active

9 Conclusion

This setup guide provides complete instructions for deploying the EMG-controlled exosuit system. Follow each step sequentially and refer to the accompanying figures for visual guidance. The system enables real-time control of exosuit movements based on EMG signal predictions from trained machine learning models.