ARDUINO CODE

```
 #include <Servo.h>

// Create 6 Servo objects
Servo servo1, servo2, servo3, servo4, servo5, servo6;

// EMG sensor pin
const int emgPin = A0;

void setup() {
  Serial.begin(9600);

  // Attach servos to their pins
  servo1.attach(9);
  servo2.attach(10);
  servo3.attach(11);
  servo4.attach(5);
  servo5.attach(6);
  servo6.attach(3);

  // Set servos to a neutral position at startup (optional)
  openHand();
}

void loop() {
  // 1. Read EMG sensor and send it to Raspberry Pi
  int emgValue = analogRead(emgPin);
  Serial.println(emgValue);
  delay(10); // Limit serial data rate

  // 2. Check if a command ("open" or "close") is received from Pi
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim(); // Remove any whitespace/newlines

    if (command.equalsIgnoreCase("open")) {
      openHand();
    }
    else if (command.equalsIgnoreCase("close")) {
      closeHand();
    }
  }
}

// Function to open hand
void openHand() {
  servo1.write(180);  // Finger 1 Open
  servo2.write(0);    // Finger 2 Open
  servo3.write(180);  // Finger 3 Open
  servo4.write(0);    // Finger 4 Open
  servo5.write(90);   // Thumb Halfway
  servo6.write(180);  // Wrist or Other movement
}

// Function to close hand
```

```
void closeHand() {
  servo1.write(0);    // Finger 1 Close
  servo2.write(180);  // Finger 2 Close
  servo3.write(0);    // Finger 3 Close
  servo4.write(180);  // Finger 4 Close
  servo5.write(0);    // Thumb Close
  servo6.write(0);    // Wrist or Other movement
}
```

NN MODEL CODE

```
import numpy as np
import tflite_runtime.interpreter as tflite
import serial
import time

# Load the TFLite model
interpreter = tflite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Connect to Arduino
arduino = serial.Serial('/dev/ttyUSB0', 9600)  # Change port if needed
time.sleep(2)  # Give Arduino time to reset

print("? System Ready. Reading EMG values...")
```

```
while True:
  try:
    # Step 1: Read EMG value from Arduino
    arduino_line = arduino.readline().decode().strip()
    print(f"Received: {arduino_line}")

    if arduino_line == "":
      print("? Empty line, skipping...")
      continue

    try:
      value = float(arduino_line)
    except ValueError:
      print("? Invalid number format.")
      continue

    if not (200 <= value <= 700):
      print("? Value out of expected EMG range (200-700).")
      continue

    # Step 2: Prepare input for model
    full_input = [value] + [value] * (50 - 1)  # Pad to 50 samples
    input_data = np.array(full_input, dtype=np.float32).reshape(1, 50, 1)

    # Step 3: Predict gesture
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output = interpreter.get_tensor(output_details[0]['index'])
    prediction = output[0][0]

    gesture = "Open" if prediction > 0.5 else "Close"
    print(f"?? Predicted Gesture: {gesture}")

    # Step 4: Send prediction back to Arduino
    arduino.write((gesture + "\n").encode())
    print(f"? Sent '{gesture}' to Arduino.\n")

except KeyboardInterrupt:
    print("?? Exiting...")
    break
except Exception as e:
    print(f"? Error: {e}")
```