1 Significance

Every year, about 700,000 Americans suffer a stroke, making it the third most frequent cause of death and the leading cause of permanent disability in the country[1]. Survivors of stroke undergo physical wrist therapy to regain functional movement for daily activities[2]. Traditional physical therapy, primarily guided by physical therapists, is a standard method for treating conditions like Stroke, Arthritis, SCI and Carpal Tunnel Syndrome [3]. However, this conventional approach often leads to therapist fatigue, potentially diminishing the quality of care and increasing patient costs [4][5]. Moreover, based on a 2018 study conducted by the World Health Organization, half of the population requiring rehabilitation encounters restricted access primarily due to the elevated expenses associated with physical therapy [6]. Excluding surgical expenses, the annual cost of robotic-assisted therapy can reach up to $70,000 for patients [7]. Thus, there is a urgent need for wrist rehabilitation robots to address these challenges.[8]

Recent clinical research highlights the benefits of robotic technology in aiding individuals with hand impairments, accelerating recovery, and lessening the burden on therapists [9][10]. Research detailed in [11] suggests that rehabilitation robotics can be integrated into specific training routines, offering a cost-effective alternative to intensive one-on-one physiotherapy sessions. Notably, robotic training has been shown to be as effective as traditional rehabilitation, with some studies indicating superior outcomes for patients undergoing robotic training compared to conventional methods[12].

Furthermore, the integration of robotics with gamification environments is beneficial[13]. This directly affects the patient's motivation and cognitive engagement, which eventually determines the efficacy of rehabilitation[14]. Findings suggest that gamification, applied in conditions such as shoulder surgery, rheumatoid arthritis, and low back pain, yields outcomes comparable or superior to conventional therapies[15][16][17].

2 Innovation

Recent developments in rehabilitation technology have greatly enhanced forearm and wrist movement exercises. Notable devices in this domain include the Cramer system[18], Rice Wrist[19], Hand Mentor[20], and Hward [21], each specifically tailored for these types of exercises. An innovative addition to this field is the Droid glove, which integrates game-based rehabilitation methods [22]. This system features a wrist rehabilitation application compatible with Android smartphones, leveraging motion and position sensors to facilitate specific hand movement exercises. It allows therapists to tailor exercises to individual needs, with the app adjusting according to the user's progress. Additionally, this system can relay movement data to doctors, aiding in the evaluation of treatment effectiveness[22]. Moreover, some rehabilitation technologies have incorporated virtual reality and mixed reality, which has shown significant benefits in improving wrist range and speed for many patients[23][24][25]. A noteworthy advancement is the combination of Leap Motion technology with a wrist rehabilitation robot and the Digits-wrist worn sensor [26][27][28]. This integration, along with elements of gamification, substantially improves the rehabilitation process[27][29], offering a compelling and effective method for patients to enhance hand functionality[30].

We designed a robotic sleeve for wrist rehabilitation, aiming to enhance hand motor functions. The lightweight and portable sleeve facilitates movement in four directions: abduction, adduction, extension, and flexion. To increase patient engagement, we have developed a game where the patient catches virtual spheres using physical wrist movements. Spheres randomly appear on screen for the patient to reach, disappearing after a set amount of time. The range that these spheres appear in is tailored to the patient's range of motion, with spheres able to spawn a bit further than the maximum distance reached in each direction. This encourages the patient to try to reach just beyond what they are able to, strengthening their muscles and range of motion. The flexibility of the game environment also allows for the therapist to create unique patterns for the patient to follow. The game can lead the patient through rehabilitation techniques, like circular or spiral patterns, with a focus on accuracy or speed, and decrease load on the therapist. An added benefit of gamification is the therapist's ability to track a patient's progress. Recording their motion and success rates enables the patient to see their progress visually over time through direct comparison between their range of motion at two different points in time.

3 Approach

3.1 Aim 1: Designing and Fabrication of the Sleeve

The design of the sleeve began with a rough sketch, aiming to create a low cost, comfortable, adaptable, and lightweight piece that could withstand the forces exerted by motors and elastic bands while remaining portable. The initial concept, shown in Figure 1, featured a sleeve extending from the knuckles to just past the elbow, covering the entire forearm. Black elastic fabric, chosen for its versatility, was sewn to fit wrist diameters from 2.8 to 3.5 inches. Motors were placed on the dorsal and median sides, with the ulnar and palmar sides controlled passively via elastic bands. Hooks were added where strings and elastic bands attached to the fabric, allowing tension adjustment as needed.

As fabrication commenced, it became evident that the design required refinement to prevent fabric from deforming. This led to the introduction of three straps between different fingers, augmenting the structure and fit. The first and second straps, sewn at both ends, are labeled 1 and 2 in Figure 2. The third strap, marked as 3 and fitted with Velcro, provided adjustable comfort. Strategically placing heat welded guides were added to direct the forces on the hand, facilitating abduction, adduction, extension, and flexion.

The sleeve's design required iterations to reinforce the softer parts of the sleeve to withstand the motor's applied forces.To prevent the motor from pulling the sleeve of the fabric down the arm and to allow wrist manipulating, a velcro band was added above the elbow, which locks the fabric of the sleeve in place. The elastic fabric for the sleeve necessitated additional support to hold the motor in place against the forearm. A rubber band is wrapped around the forearm covering the back of the motor, applying a resistive force to keep it flat.

The sleeve can be manufactured using easy to source materials for a low total cost of $101. The arduino comes in as the biggest ticket item, costing $25. The construction of the sleeve, including all fabric, pins, and elastics, costs $25. The IMUs are $12 each, and the 2 motors and drivers cost $12 together. The various electrical components, including breadboard, power supply and wiring, cost $15. A table for this data can be found in Appendix A.

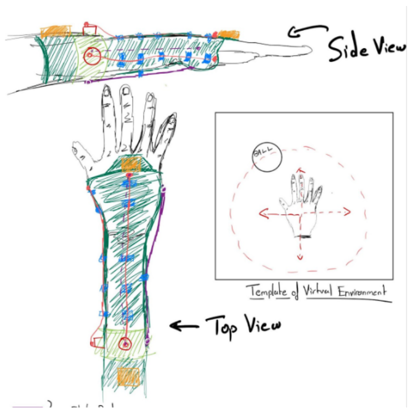

Figure 1: Interpolation for Data 1

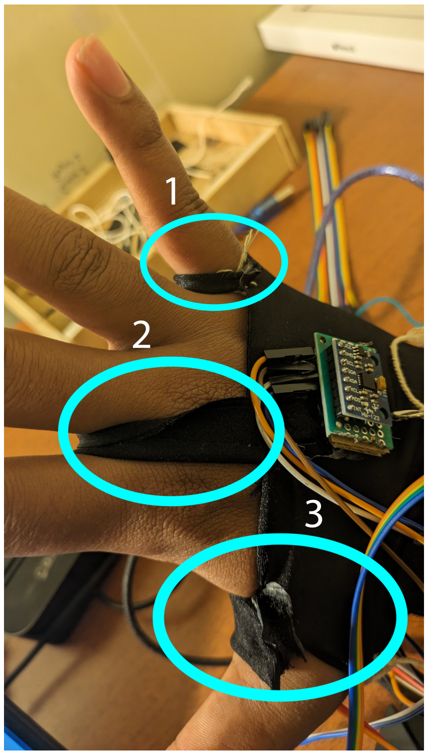

Figure 2: Interpolation for Data 2

3.2 Aim 2: Motor Integration

Aim 2: Motor Integration Two motors, specifically 24BYJ48 Stepper Motors, are strategically mounted on the dorsal and median sides of the forearm, integrating into the design. This placement is complemented by hooks on the ulnar and palmar sides, which serve to attach elastic bands, crucial for the reciprocal movement mechanism. The stepper motors are driven by a ULN2003N Stepper Motor Driver and controlled by an Atmega328P microcontroller housed on an Arduino UNO board, ensuring reliable and programmable motor operation. The mechanism operates on a simple yet effective principle: flexion is achieved passively when the motors do not exert tension, allowing the elastic bands to maintain their normal length. In contrast, extension is actively driven by the motors pulling the strings, causing the elastic bands to deform. A small amount of tension is needed from the motor that is not driven to balance out the elastic band and straighten the hand. This system is mirrored for the movements of adduction and abduction, utilizing a similar arrangement of strings, motors, and elastic bands to facilitate these, demonstrating a versatile and adaptive approach.

3.3 Aim 3: Sensor Integration and Game Development

The wrist rehabilitation system employs two Inertial Measurement Units (IMUs) to accurately monitor the wrist's angles, crucial for the precise tracking of movements. One IMU is strategically mounted at the base of the wrist, a location that remains stationary in relation to the hand, serving as a stable reference point. In contrast, the second IMU is placed near the knuckles to measure more dynamic movement. This dual-IMU setup allows for a comprehensive tracking of wrist movements, with the base IMU acting as a reference to account for any unintended full-arm movements by the patient.

To ensure the reliability of the data collected, the system includes a filtering process to eliminate any undesirable noise from the IMUs, enhancing the accuracy of the measurements. The processed data, specifically the yaw and pitch angles of the wrist, are then transmitted via a Serial bus to a computer. This computer runs an interactive game, as illustrated in Figure 3, designed to aid in rehabilitation. In the game, the patient controls a virtual hand, using their actual wrist movements, to catch spheres that appear randomly within the virtual environment. The frequency of the spheres' appearance is adjustable based on the difficulty setting, allowing for tailored rehabilitation intensity. In addition, the game can be customized for different training exercises, allowing spheres to populate in star or spiral patterns. The game is not only a tool for therapy but also serves as a metric for progress, recording their accuracy, speed, and flexibility over time. This provides valuable feedback on the patient's performance and recovery trajectory, offering an engaging and measurable way to track rehabilitation progress over time.

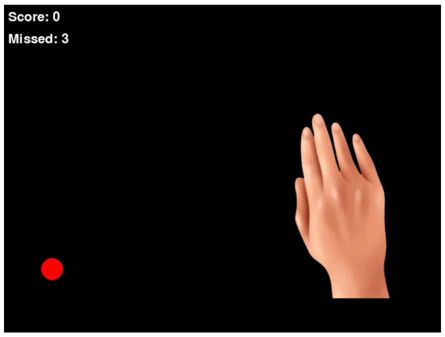

Figure 3: Game Design

References

[1] W. G. MEMBERS, D. Lloyd-Jones, R. J. Adams, T. M. Brown, M. Carnethon, S. Dai, G. De Simone, T. B. Ferguson, E. Ford, K. Furie, *et al.*, "Heart disease and stroke statistics–2010 update: a report from the american heart association," *Circulation*, vol. 121, no. 7, pp. e46–e215, 2010.

[2] J. P. P. v. V. Hussain, S. and M. Ghayesh, "State-of-the-art robotic devices for wrist rehabilitation: Design and control aspects," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 5, pp. 361–372, 2020.

[3] N. P. Behrman AL, Bowden MG, "Neuroplasticity after spinal cord injury and training: an emerging paradigm shift in rehabilitation and walking recovery.," *Physical Therapy*, vol. 86, no. 10, pp. 1406–1425, 2006.

[4] M. S. Maia, G. L. Dos Santos, and C. C. Porto, "Work-related stress among physiotherapists from the musculoskeletal area: An observational study," *Work*, vol. 71, no. 4, pp. 1163–1173, 2022.

[5] M. Okhiria, A. Truszczyńska-Baszak, and A. Tarnowski, "Assessment of work-related fatigue in polish physiotherapists and of its effect on their diagnostic accuracy and physiotherapy planning," *International Journal of Occupational Safety and Ergonomics*, vol. 26, no. 2, pp. 406–412, 2020.

[6] M. Panny, A. Mayr, M. Nagiller, and Y. Kim, "A domestic robotic rehabilitation device for assessment of wrist function for outpatients," *Journal of Rehabilitation and Assistive Technologies Engineering*, vol. 7, p. 2055668320961233, 2020.

[7] K. Nas, L. Yazmalar, V. Şah, A. Aydın, and K. Öneş, "Rehabilitation of spinal cord injuries," *World Journal of Orthopedics*, vol. 6, no. 1, pp. 8–16, 2015.

[8] F. Sergi, M. M. Lee, and M. K. O'Malley, "Design of a series elastic actuator for a compliant parallel wrist rehabilitation robot," pp. 1–6, 2013.

[9] C. Takahashi, L. Der-Yeghiaian, V. Le, R. Motiwala, and S. Cramer, "Robot-based hand motor therapy after stroke," *Brain*, vol. 131, no. 2, p. 425–437, 2008.

[10] C. L. Jones, F. Wang, C. Osswald, X. Kang, N. Sarkar, and D. G. Kamper, "Control and kinematic performance analysis of an actuated finger exoskeleton for hand rehabilitation following stroke," pp. 282–287, 2010.

[11] P. E. Byl NN, Abrams GM, "Chronic stroke survivors achieve comparable outcomes following virtual task specific repetitive training guided by a wearable robotic orthosis (ul-exo7) and actual task specific repetitive training guided by a physical therapist.," *Journal of Hand Therapy*, vol. 26, no. 4, pp. 343–352, 2013.

[12] L. C. Lo K, Stephenson M, "Effectiveness of robotic assisted rehabilitation for mobility and functional ability in adult stroke patients: a systematic review.," *JBI Database System Rev Implement Rep.*, vol. 15, no. 12, pp. 3049–3091, 2017.

[13] V. Simões-Silva, A. F. D. Mesquita, K. L. S. Da Silva, V. S. A. Quental, and A. Marques, "Gamification as upper limb rehabilitation process," pp. 243–257, 2021.

[14] J. Oblak, I. Cikajlo, and Z. Matjačić, "Universal haptic drive: A robot for arm and wrist rehabilitation," *IEEE transactions on neural systems and rehabilitation engineering*, vol. 18, no. 3, pp. 293–302, 2009.

[15] F. M. Alfieri, C. da Silva Dias, N. C. de Oliveira, and L. R. Battistella, "Gamification in musculoskeletal rehabilitation," *Current Reviews in Musculoskeletal Medicine*, vol. 15, no. 6, pp. 629–636, 2022.

[16] J. W. Then, S. Shivdas, T. S. T. A. Yahaya, N. I. Ab Razak, and P. T. Choo, "Gamification in rehabilitation of metacarpal fracture using cost-effective end-user device: A randomized controlled trial," *Journal of Hand Therapy*, vol. 33, no. 2, pp. 235–242, 2020.

[17] V. Yazdnian, A. Delavari, H. Moradi, S. Shams, M. Teymouri, and A. Rezaei, "Comprehensive and gamified rehabilitation system for upper-limb impairment treatments," pp. 445–452, 2022.

[18] S. Spencer, J. Klein, K. Minakata, V. Le, J. Bobrow, and D. Reinkensmeyer, “A low cost parallel robot and trajectory optimization method for wrist and forearm rehabilitation using the wii,” pp. 869–874, 2008.

[19] A. Gupta and M. K. O’Malley, “Design of a haptic arm exoskeleton for training and rehabilitation,” *IEEE/ASME Transactions on mechatronics*, vol. 11, no. 3, pp. 280–289, 2006.

[20] E. Koeneman, R. Schultz, S. Wolf, D. Herring, and J. Koeneman, “A pneumatic muscle hand therapy device,” vol. 1, pp. 2711–2713, 2004.

[21] C. D. Takahashi, L. Der-Yeghiaian, V. Le, and S. C. Cramer, “A robotic device for hand motor therapy after stroke,” pp. 17–20, 2005.

[22] D. Deponti, D. Maggiorini, and C. E. Palazzi, “Droidglove: An android-based application for wrist rehabilitation,” pp. 1–7, 2009.

[23] R. Boian, A. Sharma, C. Han, A. Merians, G. Burdea, S. Adamovich, M. Recce, M. Tremaine, and H. Poizner, “Virtual reality-based post-stroke hand rehabilitation,” pp. 64–70, 2002.

[24] W. M. Naqvi *et al.*, “Gamification in therapeutic rehabilitation of distal radial and ulnar fracture: a case report,” *Cureus*, vol. 14, no. 8, 2022.

[25] A. Pillai, M. S. H. Sunny, M. T. Shahria, N. Banik, and M. H. Rahman, “Gamification of upper limb rehabilitation in mixed-reality environment,” *Applied Sciences*, vol. 12, no. 23, p. 12260, 2022.

[26] O. . I. S. . B. A. . C. J. . O. I. . O. P. Kim, David Hilliges, “Digits: Freehand 3d interactions anywhere using a wrist-worn gloveless sensor.,” pp. 167–176, 2012.

[27] M. A. et al., “Gamification of hand rehabilitation process using virtual reality tools: Using leap motion for hand rehabilitation,” *IEEE International Conference on Robotic Computing (IRC)*, vol. 50, no. 5, pp. 336–339, 2017.

[28] S. Chen, H. Ma, C. Yang, and M. Fu, “Hand gesture based robot control system using leap motion,” pp. 581–591, 2015.

[29] I. Afyouni, F. U. Rehman, A. M. Qamar, S. Ghani, S. O. Hussain, B. Sadiq, M. A. Rahman, A. Murad, and S. Basalamah, “A therapy-driven gamification framework for hand rehabilitation,” *User Modeling and User-Adapted Interaction*, vol. 27, pp. 215–265, 2017.

[30] M. A. Teruel, V. López-Jaquero, M. A. Sánchez-Cifo, E. Navarro, and P. González, “Improving motivation in wrist rehabilitation therapies,” pp. 199–206, 2019.

A Appendix 1: Cost Table

Item	Quantity	Cost per Item	Total Cost
Arduino UNO	1	$25	$25
Stepper Motor + Driver Board	2	$6	$12
MPU6050 IMU	2	$12	$24
Sleeve Materials Cost	-	-	$25
Electronics Cost	-	-	$15
		Grand Total	**$101**

Table 1: Cost of Components

B Appendix 2: Low Level Code

```
#include <Wire.h>
#include <MPU6050_light.h>
#include <Stepper.h>


MPU6050 mpu(Wire);
const float STEPS_PER_REV = 32;
//  Amount of Gear Reduction
const float GEAR_RED = 64;
// Number of steps per geared output rotation
const float STEPS_PER_OUT_REV = STEPS_PER_REV * GEAR_RED;
// Defines the number of steps per rotation
const int stepsPerRevolution = STEPS_PER_OUT_REV;

// Creates an instance of stepper class
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
Stepper myStepper1 = Stepper(stepsPerRevolution, 8, 10, 9, 11);
Stepper myStepper2 = Stepper(stepsPerRevolution, 4, 5, 6, 7);
char data;


void setup() {
  Serial.begin(115200);  // Start the serial communication
  Wire.begin();

  byte status = mpu.begin();
  Serial.print(F("MPU6050 status: "));
  Serial.println(status);
  while (status != 0) {}  // stop everything if could not connect to MPU6050

  Serial.println(F("Calculating offsets, do not move MPU6050"));
  delay(1000);
  // mpu.upsideDownMounting = true; // uncomment this line if the MPU6050 is mounted upside
  mpu.calcOffsets();  // gyro and accelero
  Serial.println("Done!\n");
}
```

```
void loop() {
  if (Serial.available()) {
    data = Serial.read();
  }

  if (data == 'u') {
    // Serial.println("Up");
    myStepper1.setSpeed(10);
    myStepper1.step(stepsPerRevolution / 6);
    // delay(1000);
  } else if (data == 'd') {
    myStepper1.setSpeed(10);
    myStepper1.step(-stepsPerRevolution / 6);
    // delay(1000);
  }

  if (data == 'l') {
    myStepper2.setSpeed(10);
    myStepper2.step(stepsPerRevolution / 6);
  }
  else if(data == 'r'){
    myStepper2.setSpeed(10);
    myStepper2.step(-stepsPerRevolution / 6);
  }

  mpu.update();
  int yaw = map(int(mpu.getAngleZ()), -24, 50, 800, 0);      // Get Yaw (Z-axis rotation)
  int pitch = map(int(mpu.getAngleY()), -45, 60, 600, 00);  // Get Pitch (X-axis rotation)

  // Send yaw and pitch to the serial port
  Serial.print("Yaw: ");
  Serial.print(yaw);
  Serial.print(", Pitch: ");
  Serial.println(pitch);

  delay(100);  // Delay to control data rate
}
```

C Appendix 2: Low Level Code

```
import pygame
import sys
import serial
import time
import threading
import random
import keyboard

# Initialize Pygame
pygame.init()

# Set up the screen dimensions and title
```

```
screen_width, screen_height = 800, 600
screen = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("Virtual Hand Movement")

# Define colors
BLACK = (0, 0, 0)
RED = (255, 0, 0)

# Load the hand image
hand_image = pygame.image.load("C:/Users/anshm/OneDrive/Desktop/hand.png").convert_alpha()
# Replace 'hand_image.png' with your image file
hand_rect = hand_image.get_rect()

# Serial communication setup
ser = serial.Serial('COM3', 115200, timeout=1)
time.sleep(2)

# Variables to store the yaw and pitch values
yaw = 0
pitch = 0

   # Sphere variables
sphere_radius = 20
sphere_x, sphere_y = -100, -100  # Initialize off-screen
last_spawn_time = 0
spawn_interval = 3000
sphere_spawned = False

# Score and missed counters
score = 0
missed = 0

# Font for displaying score and missed count
font = pygame.font.Font(None, 36)

# Safe distance definition
safe_distance = 300  # Define a safe distance

# Global variables for calibration
center_yaw = 0
center_pitch = 0

# Function to calibrate the center
def calibrate_center():
    global center_yaw, center_pitch, yaw, pitch
    center_yaw = yaw
    center_pitch = pitch
    print(f"Calibrated to Yaw: {center_yaw}, Pitch: {center_pitch}")
# Function to spawn a new sphere
def spawn_sphere():
    global sphere_x, sphere_y, sphere_spawned
    while True:
        new_x = random.randint(sphere_radius, screen_width - sphere_radius)
        new_y = random.randint(sphere_radius, screen_height - sphere_radius)
        # Calculate distance to the hand
        distance_to_hand = ((new_x - hand_rect.centerx)**2 + (new_y - hand_rect.centery)**
        if distance_to_hand >= safe_distance:  # Check if it's a safe distance
            sphere_x, sphere_y = new_x, new_y
            sphere_spawned = True
            break

def read_serial():
    global yaw, pitch
    while True:
        if ser.in_waiting > 0:
            line = ser.readline().decode('utf-8').rstrip()
            if line.startswith("Yaw:"):
                yaw_val, pitch_val = line.split(", Pitch: ")
                yaw = int(yaw_val.split(': ')[1])
                pitch = int(pitch_val)

# Start the serial reading in a separate thread
thread = threading.Thread(target=read_serial)
thread.start()


# Main game loop
while True:

    if keyboard.is_pressed('up'):
        ser.write(b'u')
        while keyboard.is_pressed('up'):  # Wait for the key to be released
            pass
    elif keyboard.is_pressed('down'):
        ser.write(b'd')
        while keyboard.is_pressed('down'):  # Wait for the key to be released
            pass
    elif keyboard.is_pressed('left'):
        ser.write(b'l')
        while keyboard.is_pressed('left'):
            pass
    elif keyboard.is_pressed('right'):
        ser.write(b'r')
        while keyboard.is_pressed('right'):
            pass
    elif keyboard.is_pressed('esc'):
        ser.write(b'e')
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            ser.close()
            sys.exit()

        # Check for a specific key press to trigger calibration
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_c:  # Press 'C' to calibrate
                calibrate_center()
```

```
current_time = pygame.time.get_ticks()

# Check if it's time to spawn a new sphere
if current_time - last_spawn_time > spawn_interval:
    if sphere_spawned:
        missed += 1
    spawn_sphere()  # Call the spawn function
    last_spawn_time = current_time

# Update hand position based on yaw and pitch
# Map yaw and pitch to screen coordinates here
adjusted_yaw = yaw - center_yaw
adjusted_pitch = pitch - center_pitch
hand_rect.x = adjusted_yaw
hand_rect.y = adjusted_pitch
# print(yaw)
# print(pitch)

    # Collision detection
if sphere_spawned and hand_rect.colliderect((sphere_x - sphere_radius, sphere_y - sphe
    score += 1  # Increment score
    sphere_spawned = False  # Reset sphere spawn


# Fill the screen with black
screen.fill(BLACK)
if sphere_spawned:
    pygame.draw.circle(screen, RED, (sphere_x, sphere_y), sphere_radius)
# Display score and missed count
score_text = font.render(f"Score: {score}", True, (255, 255, 255))
missed_text = font.render(f"Missed: {missed}", True, (255, 255, 255))
screen.blit(score_text, (10, 10))
screen.blit(missed_text, (10, 50))

# Blit the hand image
screen.blit(hand_image, hand_rect)

# Update the display
pygame.display.flip()
```