# EE399: A Virtual Reality-based Hand Rehabilitation Device for Post-Stroke Therapy

By Harshada Kale under the guidance of Dr. Uttama Lahiri

November 21, 2024

**Abstract**

Stroke rehabilitation often requires intensive hand exercises to restore motor function, but traditional therapies can be repetitive and demotivating, reducing adherence (Levin et al., 2015). This project presents a gamified virtual reality (VR)-based rehabilitation device that uses a glove equipped with flex sensors to track finger movements in real-time. The data is visualized in a VR environment, where patients perform interactive exercises, such as inflating and popping a virtual balloon or triggering musical notes by finger opposition. Initial calibration ensures the system adapts to each patient's range of motion. The device promotes motivation, adherence, and engagement by integrating immediate feedback and gamification. Preliminary tests demonstrate high accuracy in finger tracking and positive user engagement. This system has the potential for remote monitoring and therapy, offering a cost-effective and accessible solution for post-stroke hand rehabilitation.

## 1 Introduction

Post-stroke hand rehabilitation is critical for restoring fine motor skills and functional independence, but traditional therapies often lack patient engagement. Virtual reality (VR) has emerged as a promising solution, offering an interactive and engaging platform to support motor recovery. Gamified rehabilitation exercises are particularly effective in improving adherence and outcomes (Laver et al., 2017). This project focuses on a VR-enabled glove with flex sensors that tracks finger movements and provides real-time feedback through interactive exercises. By translating physical hand movements into virtual actions, the device helps patients perform necessary therapeutic exercises with increased motivation. Key benefits include:

1 . Enhanced engagement through gamified tasks.

2 . Adaptability for individual patient mobility.

3 . Potential integration into remote physiotherapy programs.

## 2 Component & Specification

### 2.1 Flex sensors

The flex sensor (model number 006 24) is a resistive sensor that detects bending by varying its resistance based on the degree of flexion. In its straight position, the sensor has a base resistance of approximately 25 kΩ, which increases as the sensor bends. For this project, five flex sensors are mounted on a glove, one for each finger, to measure finger flexion during rehabilitation exercises. The sensors are lightweight, flexible, and easy to integrate into wearable systems, making them ideal for real-time motion tracking in the VR environment.

Figure 1: Flex sensor

### 2.2 Resistors

Resistors are a chief electrical component in circuits - they are used to reduce current flow and, at the same time, maintain low voltage levels to prevent short circuits and provide the required current flow in simple circuits. In our project, we have used these resistors at several points in the circuit accordingly. The resistance values used in our circuits are - 39k Ohm.

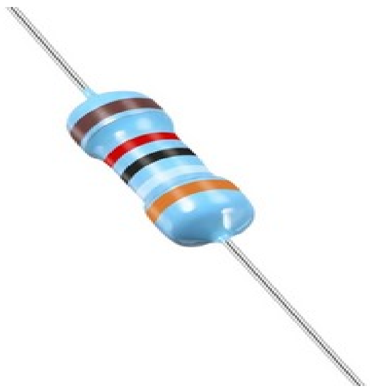
Figure 2: 39K Ohm resistor

## 2.3 Arduino Uno R3 Microcontroller

An Arduino microcontroller is a versatile tool commonly used in electronics projects due to its ability to process sensor data and control various components. For this project, I use it to read data from the flex sensors and transmit the processed values to the VR software. Using Arduino's C-based programming environment, I developed custom code to map the sensor readings to finger joint angles, enabling real-time tracking and visualization. (*code provided below*)

The Arduino Uno is a versatile microcontroller that is well-suited for electronics projects, including the VR rehabilitation device in this project. Its key specifications are as follows:

- **Analog-to-Digital Converter (ADC):** The Arduino Uno uses a 10-bit ADC, representing analog signals as digital values ranging from 0 to 1023.
- **Sampling Rate:** By default, the ADC has a maximum sampling rate of 9.6 kHz (approximately 104 µs per sample), adjustable via the ADC clock prescaler.
- **Input Voltage Range:** The analog pins measure voltage in the range of 0–5V, which can be altered using the `analogReference()` function.
- **Processing Capability:** Powered by the ATmega328P microcontroller, it operates at a clock speed of 16 MHz.
- **Memory:**
  - Flash Memory: 32 KB (0.5 KB is used for the bootloader).
  - SRAM: 2 KB.
  - EEPROM: 1 KB.
- **Connectivity and Power:**
  - 6 analog input pins (A0–A5) and 14 digital I/O pins.
  - Operates at 5V, with a recommended input voltage range of 7–12V for stable performance.
- **Special Features:** Built-in Pulse Width Modulation (PWM) functionality on 6 digital pins. Supports serial communication via USB, SPI, and I2C.

These specifications ensure precise real-time data capture and processing, making the Arduino Uno ideal for this project. For further details, refer to the official documentation available at the Arduino Official Store.

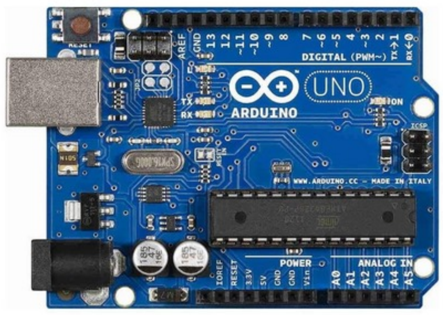

Figure 3: Arduino Uno R3 Microcontroller

## 3 Calibration

### 3.1 Initial Calibration

The 60-degree maximum bend angle was determined by measuring the flexion of the hand into a fist position across 3-4 individuals using a measuring app. The angle was consistently measured around 60°, which was then selected for the calibration process. This value was used to map sensor readings from 0° (flat hand) to -60° (fist position), providing an accurate representation of typical hand movement for rehabilitation exercises.

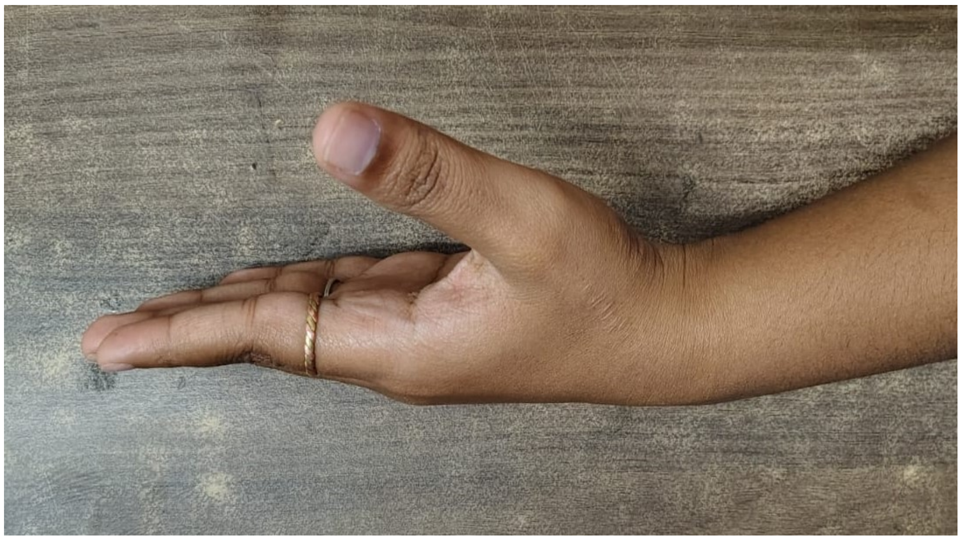

Figure 4: Initial extended fingers position

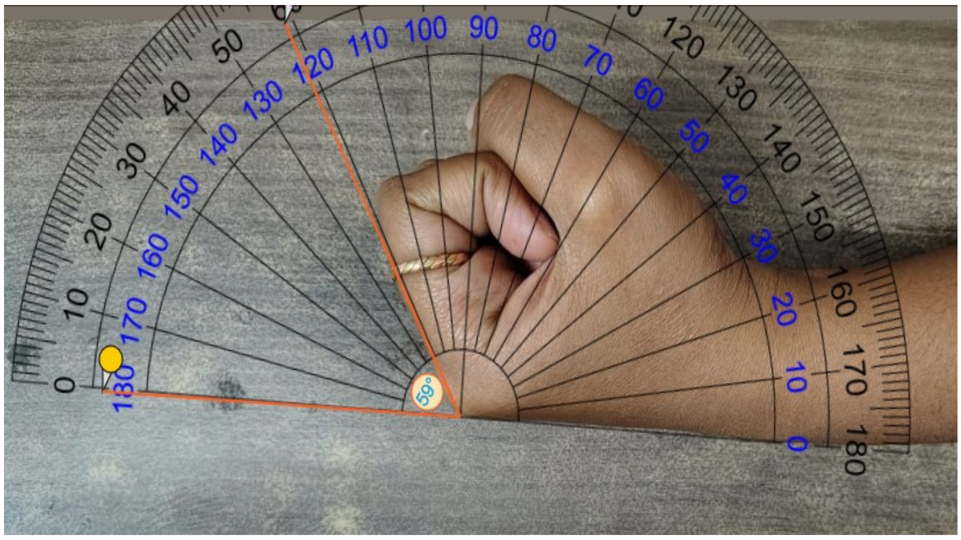

Figure 5: Maximum bend angle in a fist, measured using an online protractor

### 3.2 Patient-Specific Calibration

Calibration customizes the system to each patient's mobility by defining key positions: the flat hand position establishes a baseline (0° flexion) for all fingers, and the fist position captures the maximum flexion angle. Sensor values are then mapped to specific angles, tailoring the VR environment to the patient's range of motion. This calibration process ensures precise tracking of finger movements, enhancing the effectiveness of the rehabilitation therapy.

The calibration steps are executed as follows:

- **Step 1:** Keep the hand flat with fingers extended to record the baseline values.

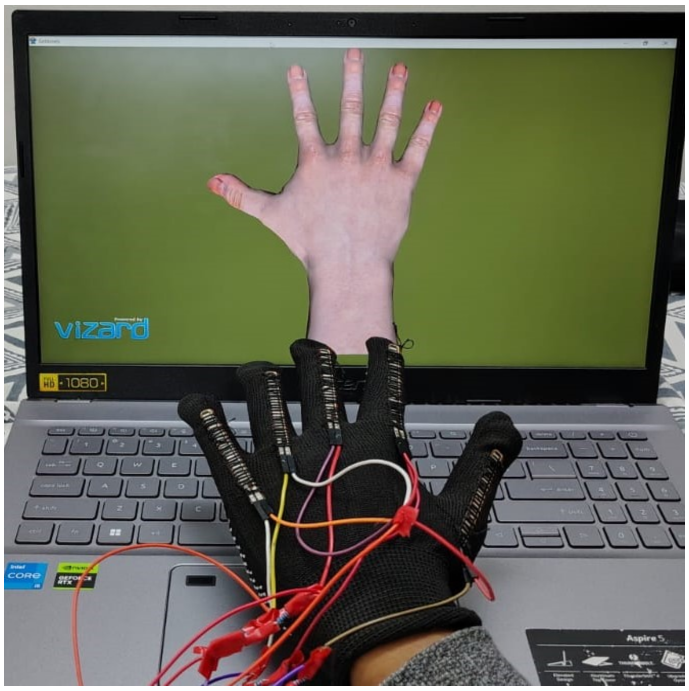

Figure 6: Shows the hand with fingers fully extended, marking the baseline for calibration.

- **Step 2:** Make a fist and hold it steady to record the maximum bend values.

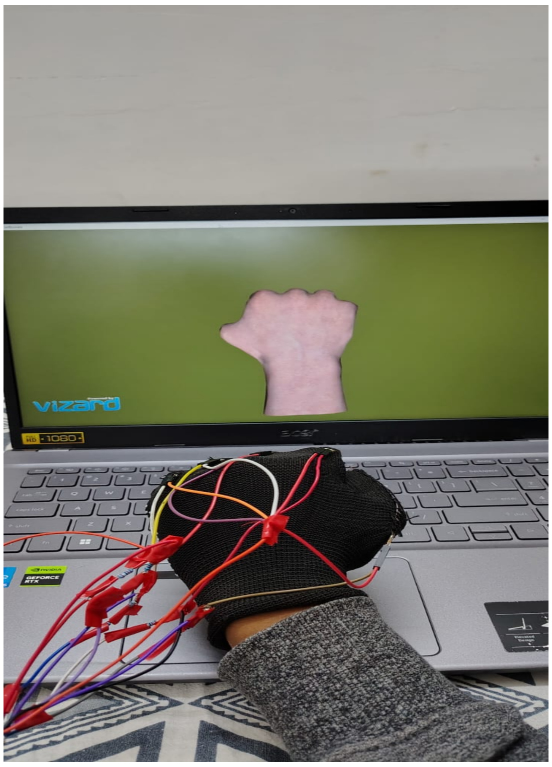

Figure 7: Shows the fist position, recording the maximum flexion angle for the calibration process.

## 4 Circuit Schematics

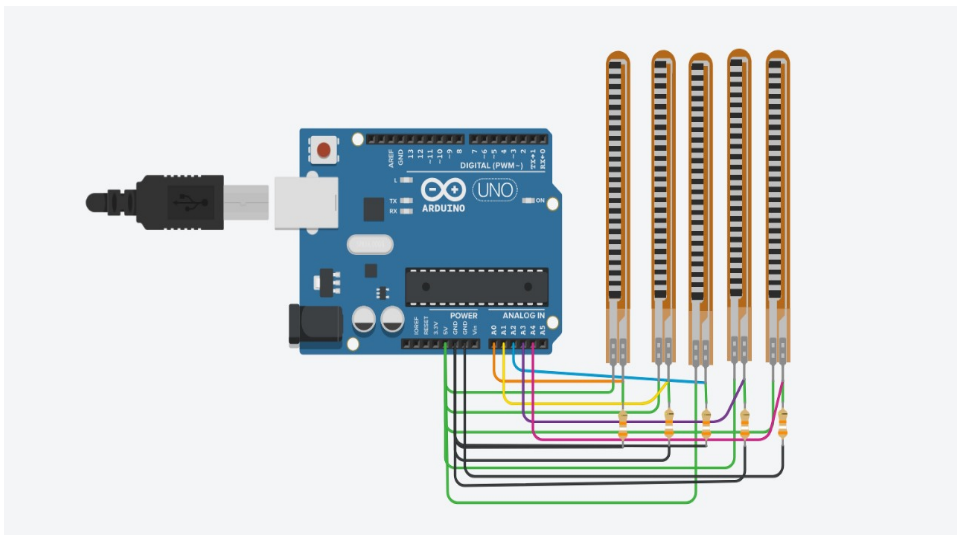

Figure 8: Circuit Design of the Flex Sensor Glove Visualized in Tinkercad

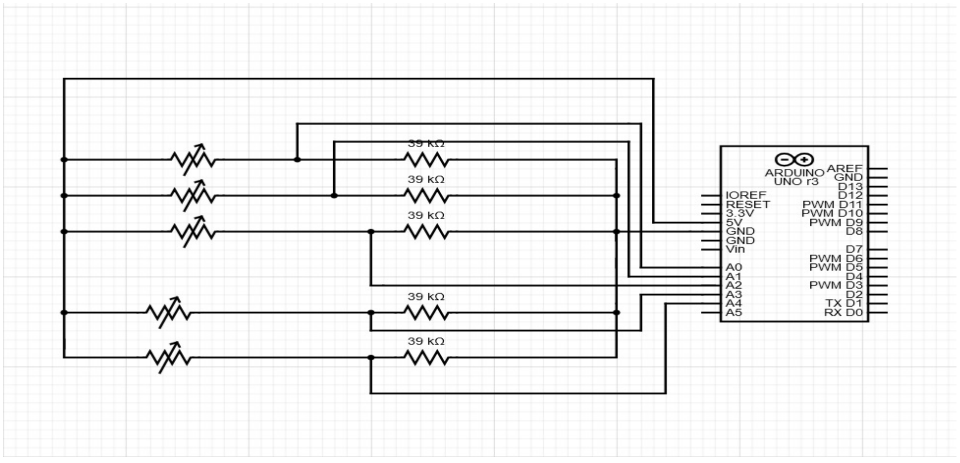

Figure 9: Circuit Diagram

The circuit consists of flex sensors connected to a voltage divider configuration, where the changing resistance of each sensor alters the output voltage. These voltages are read by the Arduino Uno's analog pins and processed to determine finger movements. A simple and efficient design ensures real-time data capture for the VR system.

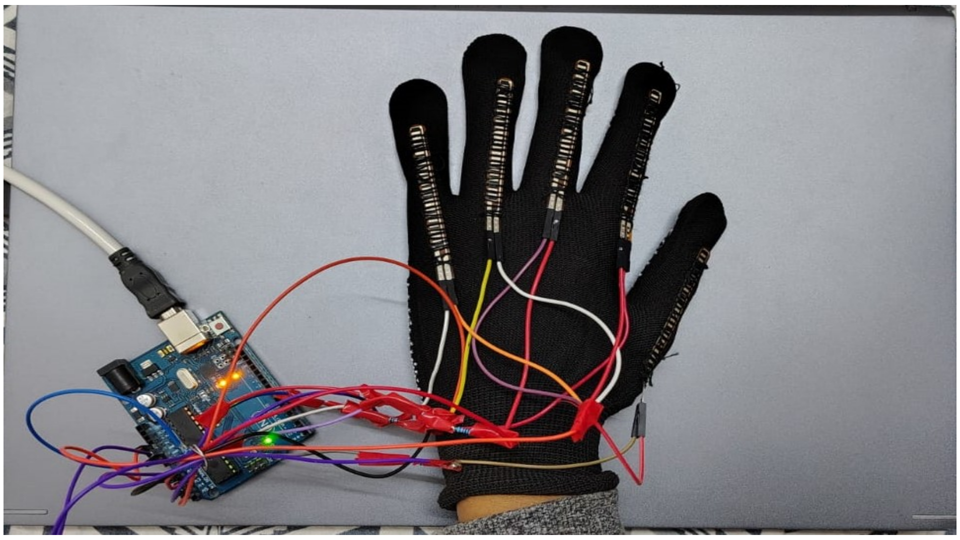

Figure 10: Circuit Diagram

## 5 Code

### 5.1 Arduino Code:

```
int flexSensorPins[] = {A0, A1, A2, A3, A4};
int flexSensorValues[5];

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

void loop() {
  for (int i = 0; i < 5; i++) {
    flexSensorValues[i] = analogRead(flexSensorPins[i]);
  }

  for (int i = 0; i < 5; i++) {
    Serial.print(flexSensorValues[i]);
    if (i < 4) {
      Serial.print(",");
    }
  }
  Serial.println();

  delay(100);
}
```

## 5.2 Vizard Code:

```
 import viz
import sys
import vizact
import time
import vizshape
import vizcam

sys.path.append(r'C:\Users\Harshada\AppData\Roaming\Python\Python311\site-packages')
import serial

viz.setMultiSample(4)
viz.fov(60)
viz.go()

day = viz.addChild('sky_day.osgb')
hand_model = viz.add('hand.cfg')
hand_model.setPosition([0, 2, 0])
hand_model.setScale([3, 3, 3])
light = viz.addLight()
light.setPosition([2, 5, 5])
light.enable()

grid = vizshape.addGrid()
grid.color(viz.GRAY)

world_axes = vizshape.addAxes()
X = viz.addText3D('X', pos=[1.1, 0, 0], color=viz.RED, scale=[0.3, 0.3, 0.3], parent=world_axes)
Y = viz.addText3D('Y', pos=[0, 1.1, 0], color=viz.GREEN, scale=[0.3, 0.3, 0.3], align=viz.ALIGN_CENTER_BASE, parent=world_axes)
Z = viz.addText3D('Z', pos=[0, 0, 1.1], color=viz.BLUE, scale=[0.3, 0.3, 0.3], align=viz.ALIGN_CENTER_BASE, parent=world_axes)

cam = vizcam.PivotNavigate(center=[0, 2, 0], distance=1)
cam.rotateRight(-30)
cam.rotateUp(15)

backgroundMusic = viz.addAudio('Music.wav')
backgroundMusic.play()
backgroundMusic.volume(0.1)

fingers = {
    "thumb": [hand_model.getBone('bone thumb 0-0'), hand_model.getBone('bone thumb 0-1')],
    "index": [hand_model.getBone('bone index 1-0'), hand_model.getBone('bone index
```

```python
    1-1'), hand_model.getBone('bone index 1-2')],
    "middle": [hand_model.getBone('bone middle 2-0'), hand_model.getBone('bone middle 2-1'), hand_model.getBone('bone middle 2-2')],
    "ring": [hand_model.getBone('bone ring 3-0'), hand_model.getBone('bone ring 3-1'), hand_model.getBone('bone ring 3-2')],
    "pinky": [hand_model.getBone('bone little 4-0'), hand_model.getBone('bone little 4-1'), hand_model.getBone('bone little 4-2')]
}

for joints in fingers.values():
    for joint in joints:
        joint.lock()

try:
    ser = serial.Serial('COM7', 9600, timeout=1)
    time.sleep(1)
except serial.SerialException:
    print("Error: Could not open serial port.")
    ser = None

def map_value(value, in_min, in_max, out_min, out_max):
    return out_min + (out_max - out_min) * (value - in_min) / (in_max - in_min)

calibration_data = {"baseline": [0] * 5, "max_bend": [0] * 5}
calibrated = False

def read_average_values(duration):
    values = []
    start_time = time.time()
    while time.time() - start_time < duration:
        if ser:
            try:
                data = ser.readline().decode().strip()
                if data:
                    flex_values = list(map(int, data.split(',')))
                    if len(flex_values) == 5:
                        values.append(flex_values)
            except ValueError:
                continue
    if values:
        return [sum(x) / len(x) for x in zip(*values)]
    else:
        return [0] * 5

def calibrate():
    global calibrated
    if not ser:
        print("Error: Serial connection not established.")
        return

    print("Calibration step 1: Keep your hand flat with fingers extended.")
    time.sleep(5)
    calibration_data["baseline"] = read_average_values(5)
    print(f"Baseline values recorded: {calibration_data['baseline']}")

    print("Calibration step 2: Make a fist and hold it steady.")
    time.sleep(5)
    calibration_data["max_bend"] = read_average_values(5)
    print(f"Maximum bend values recorded: {calibration_data['max_bend']}")

    calibrated = True
    print("Calibration complete!")

def update_fingers():
    if not calibrated:
        return

    if ser:
        try:
            data = ser.readline().decode().strip()
            if data:
                flex_values = list(map(int, data.split(',')))
                if len(flex_values) != 5:
                    print(f"Invalid sensor data: {flex_values}")
                    return

                finger_angles = {
                    "thumb": [map_value(flex_values[0], calibration_data["baseline"][0], calibration_data["max_bend"][0], 0, -60)] * len(fingers["thumb"]),
                    "index": [map_value(flex_values[1], calibration_data["baseline"][1], calibration_data["max_bend"][1], 0, -60)] * len(fingers["index"]),
                    "middle": [map_value(flex_values[2], calibration_data["baseline"][2], calibration_data["max_bend"][2], 0, -60)] * len(fingers["middle"]),
                    "ring": [map_value(flex_values[3], calibration_data["baseline"][3], calibration_data["max_bend"][3], 0, -60)] * len(fingers["ring"]),
                    "pinky": [map_value(flex_values[4], calibration_data["baseline"][4], calibration_data["max_bend"][4], 0, -60)] * len(fingers["pinky"])
                }

                for finger, angles in finger_angles.items():
                    for i, joint in enumerate(fingers[finger]):
                        if i < len(angles) and i < len(fingers[finger]) - 1:
                            if finger == "thumb":
                                joint.setEuler([0, 0, angles[i]])
                            else:
                                joint.setEuler([angles[i], 0, 0])


        except (ValueError, serial.SerialException) as e:
            print(f"Error reading serial data: {e}")


calibrate()

vizact.ontimer(0.1, update_fingers)
ball = vizshape.addSphere(radius=0.4)
ball.setPosition([0, 1, 2])
ball.color(viz.RED)
ball.visible(False)

shadow = viz.addChild('shadow.wrl', alpha=0.7, cache=viz.CACHE_CLONE)
viz.link(ball, shadow, mask=viz.LINK_POS).setPos([None, 0, None])
shadow.visible(False)

air_gush_sound = viz.addAudio('Air_gush.wav')
balloon_pop_sound = viz.addAudio('Balloon pop.wav')

exercise_active = False
balloon_scale = 0.4

def start_exercise():
    global exercise_active, balloon_scale
    exercise_active = True
    balloon_scale = 0.4
    ball.visible(True)
    shadow.visible(True)
    ball.setScale([balloon_scale, balloon_scale, balloon_scale])
    print("Balloon popping exercise started!")
    text_2D_screen = viz.addText('Pop the balloon',parent=viz.SCREEN)
    text_2D_screen.setPosition(0.3,0.9)
    text_2D_screen.color(viz.YELLOW)

    text_2D_screen.setBackdrop(viz.BACKDROP_RIGHT_BOTTOM)
    text_2D_screen.setBackdropColor([0.5,0.25,0])
    text_2D_screen.font('Times New Roman')

def update_exercise():
    global exercise_active, balloon_scale

    if not exercise_active or not calibrated or not ser:
        return

    try:
        data = ser.readline().decode().strip()
        if data:
            flex_values = list(map(int, data.split(',')))
            if len(flex_values) != 5:
                return

            thumb_bend = map_value(flex_values[0], calibration_data["baseline"][0],
    calibration_data["max_bend"][0], 0, -80)
            if thumb_bend <= -70:
                balloon_pop_sound.play()
                ball.visible(False)
                shadow.visible(False)
                exercise_active = False
                print("Balloon popped!")
                return

            for i, finger in enumerate(["index", "middle", "ring", "pinky"], start=1)
    :
                bend = map_value(flex_values[i], calibration_data["baseline"][i],
    calibration_data["max_bend"][i], 0, -80)
                if bend <= -70:
                    if balloon_scale < 1.2:
                        balloon_scale += 0.1
                        ball.setScale([balloon_scale, balloon_scale, balloon_scale])
                        air_gush_sound.play()
                        print(f"Balloon size increased to {balloon_scale:.2f}")
                    break
    except (ValueError, serial.SerialException) as e:
        print(f"Error reading exercise data: {e}")

vizact.onkeydown('1', start_exercise)
vizact.ontimer(0.1, update_exercise)
```

## 6 Results

Two interactive exercises were implemented in the VR environment to promote hand movement:

- **Balloon Pop Exercise:** This task encourages coordinated finger flexion as the patient bends their fingers (index to pinky). A virtual ball inflates gradually with each bend, and when the thumb bends past a set angle, the ball ”pops” with an accompanying sound effect. This exercise provides a therapeutic challenge and a reward mechanism for task completion, making it engaging and motivating. The Balloon Pop Exercise was successfully implemented and is designed to reinforce proper hand movement and coordination.

- **Musical Fingers Exercise:** In this task, patients are instructed to touch their thumb to each finger sequentially, triggering a unique musical note with each successful touch. This exercise aims to enhance fine motor control and finger coordination, especially after events like stroke, and helps practice finger opposition. Each finger (index to pinky) produces a different note, making the exercise engaging while also being highly beneficial for rehabilitation.

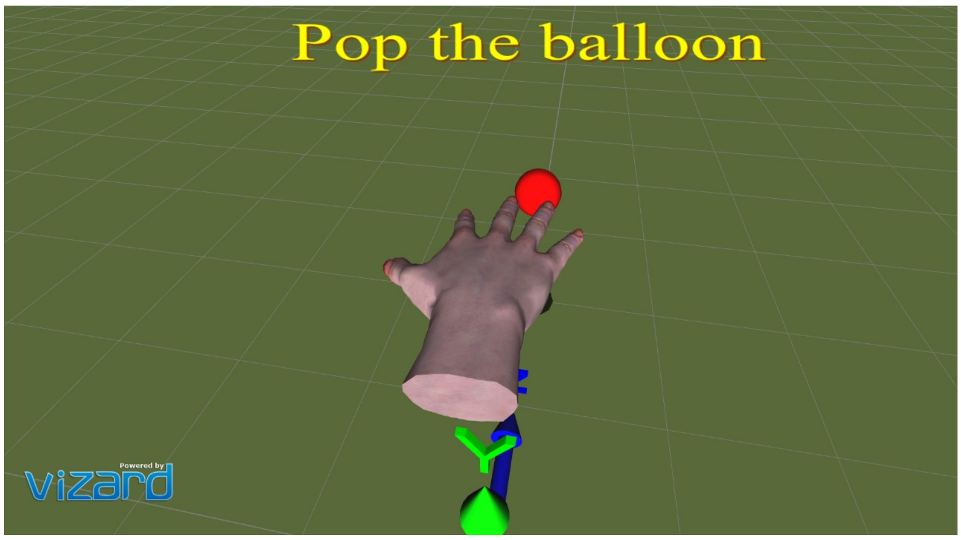

Figure 11: Balloon Pop exercise

You can view the working video by clicking the following link:
Working Video
While I successfully implemented the Balloon Pop Exercise, I began working on the Musical Fingers Exercise. To calibrate and establish the relationship between the thumb joint angles, I manually adjusted the yaw, pitch, and roll values of the thumb joints. By experimenting with different positions and mapping the corresponding sensor values, I aimed to interpolate and identify the best relation between these joint angles. Although this calibration process is not fully completed, it provides a foundation for further refinement. Alternatively, I planned to introduce another flex sensor on the palm to enhance the tracking accuracy. These exercises demonstrate the potential of VR for immersive, motivating hand rehabilitation, where patients receive immediate feedback to support adherence to therapy and promote correct movements.

## 7 Summary

This project introduces a VR-based rehabilitation glove designed to enhance post-stroke therapy. By gamifying hand exercises and providing real-time feedback, the device improves patient motivation and adherence, key factors in effective recovery. Exercises such as ”Balloon Pop” and ”Musical Fingers” target finger flexion and opposition, promoting fine motor skill development. Calibration ensures the system adapts to individual mobility, while the VR environment offers an engaging and immersive therapy experience.

## 8 Limitation

The system has several limitations, including sensor drift, where prolonged use can lead to a degradation in sensor accuracy. It also has limited tracking capabilities, focusing only on finger flexion without capturing wrist or hand orientation. Additionally, the system's performance is dependent on regular recalibration to maintain accuracy, as the sensors may shift over time or with repeated use. These factors can impact the overall effectiveness and precision of the rehabilitation experience.

## 9 References

1 Levin, M. F., Weiss, P. L., Keshner, E. A. (2015). Virtual reality and motor rehabilitation. *Springer.*

2 Laver, K. E., George, S., Thomas, S., Deutsch, J. E., Crotty, M. (2017). Virtual reality for stroke rehabilitation. *Cochrane Database of Systematic Reviews*, (11).

3 Arduino, ”Arduino Uno Rev3 Technical Specifications,” *Arduino*, Accessed: Nov. 21, 2024. [Online]. Available: `https://docs.arduino.cc/hardware/uno-rev3/#tech-specs`