Proceedings of the 9th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'23)
Brunel University, London, United Kingdom - August 03-05, 2023
Paper No. CIST 145
DOI: 10.11159/cist23.145

Real-time Interfacing of a Pneumatically-actuated Finger-thumb Rehabilitation Device

Narges Ghobadi[1*], Mohsen Khajoee[1*], Witold Kinsner[2], Tony Szturm[3], Nariman Sepehri[1]
[1]Department of Mechanical Engineering, Price Faculty of Engineering,
[2]Department of Electrical & Computer Engineering, Price Faculty of Engineering
[3]College of Rehabilitation Sciences, Rady Faculty of Health Sciences
University of Manitoba, Winnipeg, Manitoba, Canada
ghobadin@myumanitoba.ca, khajoeem@myumanitoba.ca, Witold.Kinsner@umanitoba.ca, Tony.Szturm@umanitoba.ca, Nariman.Sepehri@umanitoba.ca
[*]These authors contributed equally.

Abstract - Rehabilitation of manual dexterity of patients with sensory motor impairments of the upper extremity due to stroke requires task-specific and repetitive exercises to facilitate recovery of a function. Given that patients require a considerable amount of time in therapy, it is essential that the rehabilitation devices are engaging and game-based to prevent the boredom of long-term repetitive task practice exercise regimes. In this study, an easy-to-use finger-thumb mechanism is designed that supports the index and middle fingers, and the thumb for patients with hand injuries. The device is connected to a game wirelessly, allowing patients to interact in real time with engaging computer games using goal-directed thumb and finger movements. The connection is established using two Raspberry Pi boards with the aid of a server-client network. The device also provides assistive-resistive forces during the game to assist or challenge patients depending on the state of their motor control. The slight delay of 10 ms for data transfer and 50 ms for game event updates enables the patients to play the game and modify the game events in real time, using the wearable device.

***Keywords*:** Real-time interfacing; edge computing; rehabilitation; pneumatic actuation

1. Introduction

After a stroke, any individuals exhibit impairments in the sensorimotor control of their hand, which can significantly impact their independence and quality of life [1]. To address this issue, the use of robotic systems in post-stroke rehabilitation is gaining popularity [2] due to their abilities to offer the required intensity, motivation, and customization. Moreover, rehabilitation manipulandum can serve as motion guides, facilitating a large number of repetitions and improving hand control during reaching movements [3].

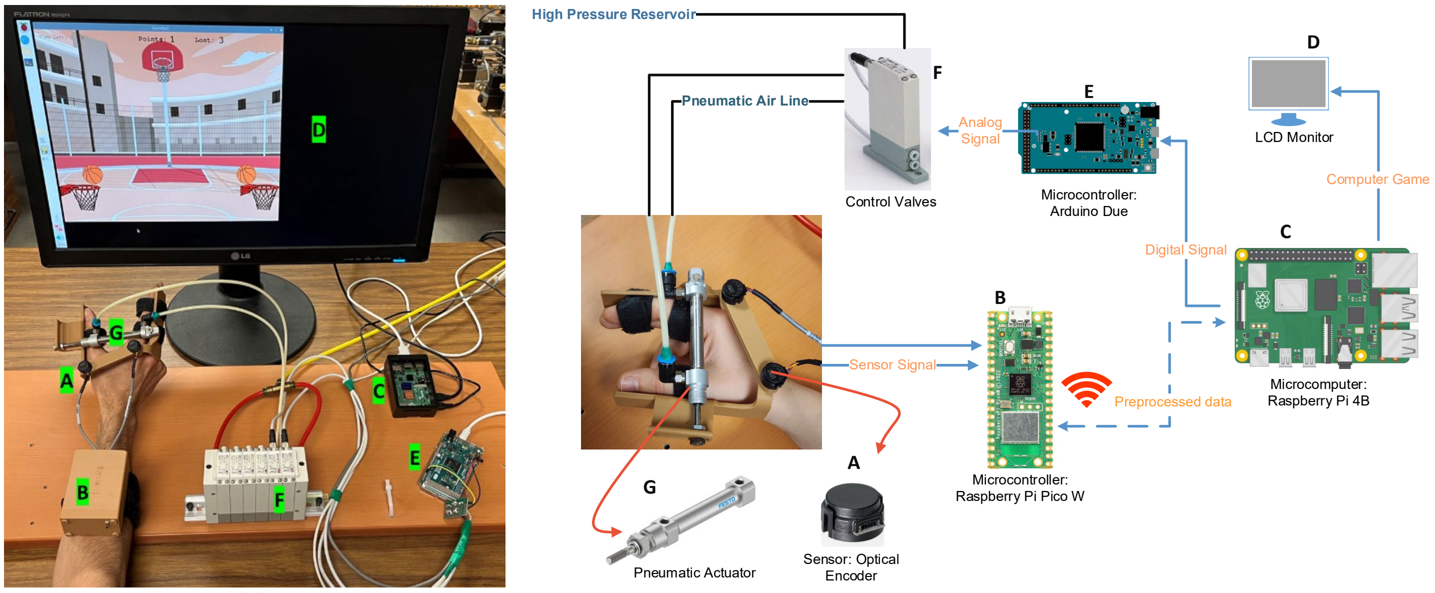
To alleviate the tedious and challenging nature of rehabilitation sessions for some patients, an effective stroke rehabilitation program should also employ an integrative approach that addresses both motor and cognitive functions. One approach to achieving this goal is to link rehabilitation devices to computer games, which not only enhances rehabilitation outcomes, but also makes the process more appealing for patients [4, 5]. However, the implementation of computer games for rehabilitation is a complicated process and entails comparing online data from both the game and the device, as well as transmitting commands to allow the device to become interactive [6]. According to the findings in the article on game latency, a simple one-dimensional game typically has a maximum acceptable delay of 100 milliseconds (ms). Any delay that exceeds this threshold has the potential to impede the player's ability significantly to respond to the game's events in real time [7]. This observation emphasizes the crucial importance of minimizing latency in playing games to ensure that the user experience is satisfactory. The other end of the speed constraints is defined by 10 ms, the average muscular response time of a person [8].

One way to improve the user's overall gaming experience is to incorporate location-based and proximity-based features into computer-augmented games. In terms of game input, "Implicit Player Input" is identified as one of the key features [9]. This allows players to send input to sensors attached to their bodies, affecting the course of the game. Therefore, body control becomes a critical element of gaming. While finger and hand movements can be detected and transmitted wirelessly to a computer through radiofrequency (RF) communication [10], only specific hand gestures can be detected. Furthermore, the method is not effective in capturing the complete states of the fingers and thumb because the use of accelerometers as sensors has limitations.

The objective of this paper is to introduce a new wearable device for finger-thumb rehabilitation. The device is designed to target both motor and cognitive functions and is connected to a computer game via a server-client network utilizing two Raspberry Pi development boards. The paper presents the hardware and software components of the system and its capabilities pertaining to real-time gaming in the context of rehabilitation objectives.

2. System Hardware

The entire experimental setup is shown in Fig. 1(a), and the setup components are presented in Fig. 1(b). Regarding the setup components, a rotary incremental encoder (A) is utilized for measuring the angle of each joint. Data acquisition from the sensors, preprocessing, and data transmission to the main processor via Wi-Fi are performed by a Raspberry Pi Pico microcontroller (B). A Raspberry Pi 4B microcomputer (C) is also responsible for running the game and transmitting signals to the valves that control the actuator's force. Moreover, an LCD monitor (D) is connected to the microcomputer, providing a graphical interface for the user to interact with the game. To convert the digital signal from the Pi 4B to an analog signal, an Arduino Due board (E) is used, and a control valve (F) is also employed to regulate the pressure in the chambers of the actuator, and a pneumatic actuator (G) provides assistive or resistive forces for the patient. The components of this setup are described in a greater detail next.



(a) (b)

Fig. 1: (a) Experimental setup, (b) Setup components.

This device design ensures the necessary range of motion (ROM) for the thumb and fingers. The ROM calculation led to the selection of a pneumatic actuator with a 30 mm (about 1.18 in) stroke (Festo Round cylinder DSNU-S-8-30-P-A-MQ). This actuator can generate forces up to 22.6 N and 30.2 N during retraction and extension, respectively.

Since the mechanism operates using rotary joints, rotary encoders are used to measure the angles of the joints. Two rotary encoders are used on each joint to detect both thumb and finger motions and obtain the hand's state. The US Digital E16 micro-optical kit encoder has been selected to obtain digital quadrature signals for high-volume and restricted-space applications. The sensor is powered by 5V supply, consuming 16 mA. It has a maximum output frequency of 1.6 MHz and 80 ns output fall and rise time. This optical sensor offers 2000 counts per revolution (CPR) and a precision of 0.18 degrees for angle measurement. Although gyro sensors are capable of measuring acceleration and velocity, they are not an ideal option for measuring instantaneous angular/linear position due to two reasons. Firstly, they tend to drift as the angle is calculated indirectly via integration. Secondly, integration creates a delay in the system which is unacceptable for real-time applications.

The device is controlled using a combination of two microcontrollers, namely a Raspberry Pi 4 Model B (4GB) board and a Raspberry Pi Pico W, which enhances the reliability of the system. The former is equipped with a 64-bit Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) processor, and 4GB LPDDR4-3200 SDRAM. It is running the Raspberry Pi OS which is based on Debian operating system. Meanwhile, the latter microcontroller is an RP2040 chip designed by Raspberry Pi. It is composed of a dual-core Arm Cortex-M0+ processor capable of running up to 133 MHz, 264KB of SRAM (static RAM), and 2MB of flash memory.

The encoder is connected to the Raspberry Pi Pico, where the acquired data undergoes preprocessing before being transmitted to the Raspberry Pi 4. Based on the transmitted data, the actuator generates assistive/resistive forces during the game for the patient. It is noteworthy that the valves supplying the air flow to the actuator are controlled by analog signals from an Arduino Due board, which is based on the Atmel SAM3X8E ARM Cortex-M3 CPU.

As previously stated, the system employs two separate boards, each of which is outfitted with a Wi-Fi module. The Raspberry Pi Pico is attached to the patient's forearm and functions as part of the apparatus. Due to the wireless connection facilitated by the two boards, the patient can move or walk around while wearing the device without fear of disrupting the connection by damaging the wires. Had only one board been utilized, the patient would have been unable to move around freely or increase the distance from the screen, since the Raspberry Pi 4 must be connected to all interfaces, including the LCD monitor.

3. System Software

System programming is performed using the Python programming language, enabling easy access to various modules and libraries. To achieve real-time interfacing, several steps are executed, as outlined in the following. The overall process is illustrated in Fig. 2.

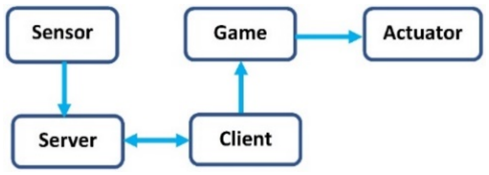


Fig. 2: Data handling diagram.

3.1. Sensor and Actuating Force

Two incremental encoders are used to measure the angle of the joints of the mechanism. The selection of incremental encoders for the mechanism was due to precision and dimensional limitations. However, it should be noted that these encoders are only capable of measuring displacement rather than absolute position. As a result, the mechanism must be in a specific state, known as the reference state, when the sensors are powered. To address this issue, the index signal is utilized to detect the reference position. The index pulse produces a single pulse during each shaft rotation, indicating a fixed and discrete position in the mechanical rotation of the encoder. Although the index pulse is commonly used as a counter to track the number of shaft rotations, it is employed in this context to measure the reference position and obtain the precise state of motion during opening and closing movements. Thus, when Index 1 (or Index 2) is stimulated, the initial rotation angle of the first link (or second link) will be set at zero degrees, serving as a benchmark. Subsequent variations in the link's rotation angle are subsequently measured relative to this point of reference.

When rotating, incremental encoders emit pulses (orthogonal signals) that must be counted to measure angular displacement accurately. Failure to detect these signals results in inaccurate measurements. As a result, it is critical to read signals from sensors at a significantly high frequency. One approach to achieving this high frequency is to allocate one of the processor cores of the microcontroller to read the signals from the sensors continuously using multi-thread coding. This method, however, is not particularly efficient in terms of processing, and the microcontroller may become overloaded. Alternatively, the interrupt request (IRQ) method is employed. Whenever there is a change in the sensor signals, an interrupt signal is transmitted to the processor to increase or decrease the counter and obtain the angular position of the joints. Therefore, the IRQ is only dispatched to the processor upon detecting any alteration in the index signal, leading to the execution of the corresponding commands. Furthermore, in addition to detecting the index signal, the IRQ is employed for both rotational directions, thereby enabling both increment and decrement of the angle. The encoder is connected to the Raspberry Pi Pico WH and programmed using MicroPython, which is the primary programming language for the board.

The noise in the system that can affect the encoder output is categorized as follows:

(i) Electrical noise is caused by electrical interference in the encoder signal, such as from electromagnetic radiation or other nearby electrical devices. As the output signal of the encoder is digital, it can tolerate a certain level of noise without significantly affecting the interpreted data from the sensor. Therefore, electrical noise can be neglected as long as the sensors are not located in an environment with strong electromagnetic radiation or other nearby high-voltage electrical devices. To mitigate noise, wires from the sensors are typically shielded and soldered to the circuit board to establish secure wire connections and minimize the impact of noise.

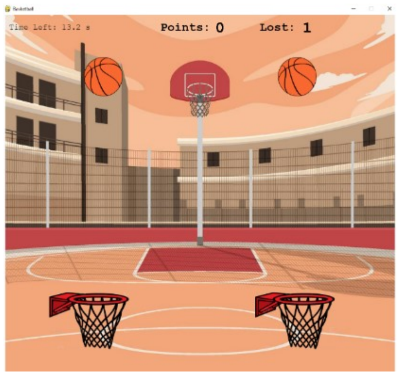
(ii) Mechanical noise may occur due to vibrations or other mechanical problems such as backlash in the shaft and encoder system. To enhance the accuracy and reliability of the encoder output, the mechanical design and tolerances have been improved, and multiple 3D printing iterations have been carried out.

To provide assistive or resistive force during game events, a pneumatic actuator is installed on the device. Valves supply air to the actuator, and these valves receive an analog signal via an Arduino Due board. To ensure that the valves receive the appropriate voltage corresponding to the game event, Python is employed to control the serial monitor window of the Arduino board.

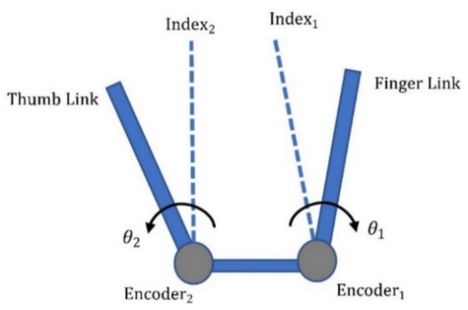
3.2. Gaming Environment

In this study, a computer game has been developed using the Pygame library. The game involves a pair of balls symmetrically appearing at equal distances from the central vertical axis at the top of the screen and moving downwards at constant speed. The user's objective is to adjust the distance between the two baskets by moving thumb and finger, to allow the balls to pass through them. The game graphical user interface is depicted in Fig. 3(a), while the encoders rotation is mapped to the distance between the two baskets, as illustrated in Fig. 3(b). During gameplay, the position of the left ball and left basket is recorded at a frequency of 10 Hz. At the end of the game, the recorded data are used to plot the distance between the balls and baskets over time. This allows for the analysis of the player's performance and provides insight into patients' progress.

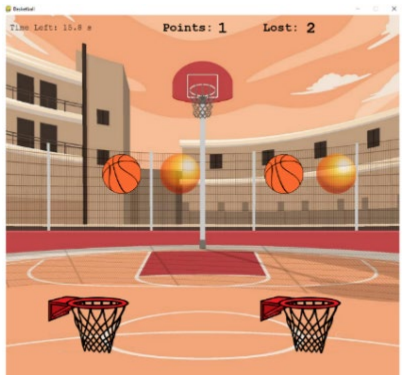
The proposed game consists of four distinct modes, each offering a different gameplay experience. The regular mode requires the user to play the game by relying on the movements of their thumb and fingers solely, without any external force. In the distraction mode, a pair of translucent balls appear on the screen alongside the original balls (Fig. 3(c)), and the user is challenged to identify the original balls while ignoring the distraction balls. No actuation force is provided in this mode. The assistive mode involves the actuator exerting force in a direction that helps the user move the balls towards the baskets, thereby facilitating gameplay. On the other hand, the resistive mode presents a challenge to the user by requiring them to overcome the force exerted by the actuator in a direction to move the balls away from the baskets. These four modes offer a variety of gameplay experiences and can be selected based on the user's preference and level of skill.



(a)



(b)



(c)

Fig. 3: (a) Basketball game, (b) Mechanism schematic, (c) Basketball game with distraction balls.

3.3. Data Transfer

As stated earlier, the encoder data are preprocessed. This involves summing the absolute values of the two rotation angles depicted in Fig. 3(b) to derive the overall rotation angle of the mechanism by the Raspberry Pi Pico. To transfer preprocessed data to the Raspberry Pi 4 and map it to the game event in subsequent steps, these two boards need to be connected either through a wired or wireless connection. Since this device is intended to be used easily, a wireless connection connection is preferred to enable patients to move freely during training without the interference of extra wires. To achieve this objective, a server-client protocol is employed, with the Raspberry Pi 4 serving as the client and the Raspberry Pi Pico serving as the server. The connection is established using 2.4 GHz 802.11n wireless LAN supported by both Raspberry Pi boards. Subsequently, the received data are mapped into the game, and game events change accordingly. To accomplish this task, a separate thread is defined for the game with the aid of multi-thread coding. This enables the game to run separately, preventing any disruption to the data transfer and minimizing system delay. In this regard, in the microcomputer, Raspberry Pi 4B, the game and data transfer processes are executed concurrently in separate cores of the processor. Due to the game screen's specific update frequency and the distinct frequency used for data transfer, executing both processes sequentially in the same loop would result in reduced frequency for both the game and data transfer. This would significantly increase the system delay.

Previously mentioned software component frequencies are as follows: Encoder at 1.6 MHz, IRQ at 33 MHz, Wi-Fi at 2.4 GHz, and Arduino serial communication at a baud rate of 9600, equivalent to 9.6 kbps (single-bit transmission).

4. Discussion of Results

Four distinct modes of device operation are examined: (i) regular game, (ii) game with distraction, (iii) game with assistive force, and (iv) game with resistive force. The games are played for 25 seconds during which, each line in the plot depicted in Fig. 4(a), represents the position error from the moment the ball appears on the screen until it disappears (either in the baskets or lost). The speed of the game remained constant throughout the gameplay, resulting in similar durations of movement for the balls, with some exceptions for the longer movements of the lost balls (which move until the end of the screen). The analysis revealed that for the regular mode game it took approximately 2.5 seconds for the player to position the basket correctly to catch the ball. In this plot, ball 3, for instance, is a lost ball due to its higher error.

$$error = X_{ball} - X_{basket} \tag{1}$$

in which X_{ball} and X_{basket} are the distances of the balls and baskets from the central vertical axis in the game, respectively.

The movement frequency of the mechanism is dependent on the speed of the balls within the game, as players must adjust the mechanism to capture them. In the normal speed mode utilized in these tests, the balls take approximately 4.2s to descend to the height of the baskets. Given that the mechanism moves from the fully closed state to the fully opened state, or vice versa, during each game event, the period of the input signal (i.e., finger-thumb movement) is 8.4s.

The angular position of each link, θ, and its maximum derivative are presented in (2) and (3) respectively:

$$\theta = \frac{FS}{2}\sin\left(\frac{2\pi}{T}t\right) \tag{2}$$

$$\max\left(\frac{d\theta(t)}{dt}\right) = \frac{FS\pi}{T} \tag{3}$$

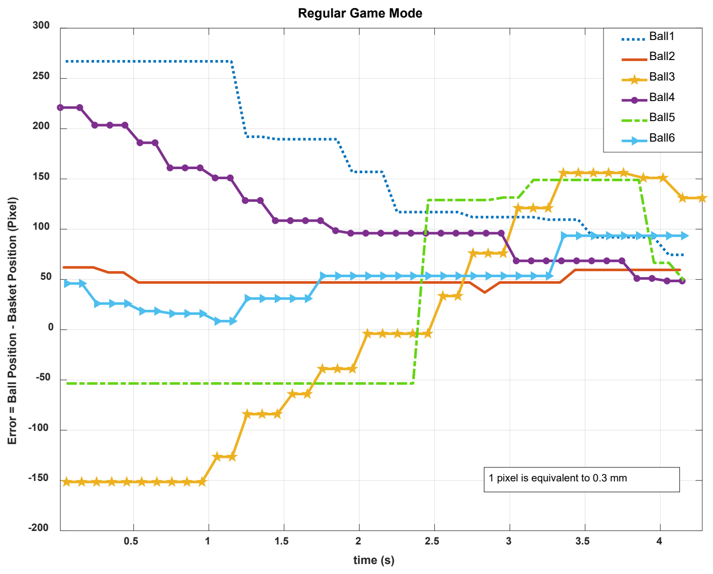
where FS is the amplitude of the signal. The calculation of the aperture time, T_a, is derived from (4) [8]. To achieve a seamless interaction between the mechanism and the game, quantization step, Q, must adhere to $Q = \frac{FS}{10^6}$. Consequently, the sampling frequency, $f = \frac{1}{T_a}$, must be no less than the value expressed in (5).

$$T_a \leq \frac{Q}{max\left(\frac{d\theta(t)}{dt}\right)} = \frac{\frac{FS}{10^6}}{\frac{FS\pi}{T}} = 2.67 \times 10^{-6} \tag{4}$$

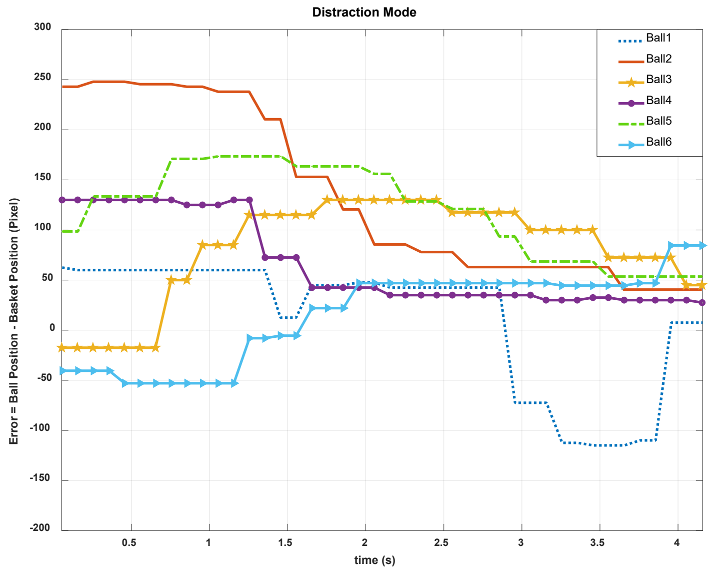
$$f \geq \frac{1}{2.67 \times 10^{-6}} = 0.374\ MHz \tag{5}$$

Since the data are acquired with the maximum output frequency of the sensor, i.e., 1.6 MHz, the sampling frequency satisfies (5). Therefore, the analog input signal can be reproduced with an exceedingly small quantization error. It should noted that the game speed can be adjusted and increased up to 4 times faster. In that case, repeating the above calculations leads to the minimum sampling frequency of 1.496 MHz which is lower than the sampling frequency of the encoder.

Figure 4(b) shows the game involving distraction balls. The findings indicate that it took approximately 3 seconds for the participant to differentiate between the authentic balls and the distraction balls, leaving only one second to maneuver the basket to the intended location. This limited period resulted in a higher degree of positional error.



(a)



(b)

Fig. 4: Position error in, (a) Regular mode, (b) Distraction mode. (1 pixel is equivalent to 0.3 mm on the screen.)

For assistive and resistive simulation modes, the status of the actuator chambers changes based on the error throughout the game. The Arduino Due's DAC0 and DAC1 pins output analog voltages with a resolution of 8 bits, this implies that they can represent 2^8 (256) different voltage levels between 0V and the reference voltage (typically 3.3V). In the presented test, two control valves are utilized with voltage levels of 110 and 220, where the higher voltage level corresponds to the valve that provides assistance or resistance during retraction (closing). To determine the output voltage produced by the DAC0 (or DAC1) pin for a given digital value, (6) can be utilized. The voltage that corresponds to the 110 voltage level is calculated where V_{out} is the output voltage of the DAC0 pin, V_{ref} is the reference voltage of the Due and D is the digital 8-bit value. By employing (6) for the 220 voltage level, the resultant output voltage is also computed as 2.7 V.

$$V_{out} = \frac{V_{ref}\ D}{256} = \frac{3.3 \times\ 110}{256} = 1.4\ V \tag{6}$$

Enabling the actuator force in the assistive mode led to the accurate positioning of the basket within 2 seconds, as depicted in Fig. 5(a). The positional errors are also found to be lower compared to the errors observed in the two preceding modes. The reason for this can be credited to the implementation of the pneumatic actuator, which aids the player in minimizing the error by utilizing feedback to control the actuator in response to the calculated error.

The resistive mode plot of Fig. 5(b) demonstrates that the player requires more time to position the basket near the desired location, and the positional error is notably higher. This is because the actuator force impedes the player's ability to manipulate the device freely, which makes it more difficult to overcome the applied force and move the baskets.

The control valves in the proposed system are connected to a constant pressure source of 32 psi. To control the actuation of the valves, the Arduino board provides either 2.7 V or 1.4 V for each control valve according to the digital values and calculations based on (6). These voltage values can be customized based on the level of impairment of the patient. The voltage values of 2.7 V and 1.4 V correspond to pressures of 8.96 and 17.54 psi, respectively, in each chamber. As a result, the actuator is capable of providing approximately 3.7 N and 1.5 N forces for the extension and retraction, respectively. This level of actuation is expected to be sufficient to facilitate the targeted physical therapy intervention. Figs. 5(c) and 5(d) depict the actuator force during gameplay for each ball from appearance on the screen until disappearance. Based on the error of the system, assistive or resistive forces are provided by the actuator.

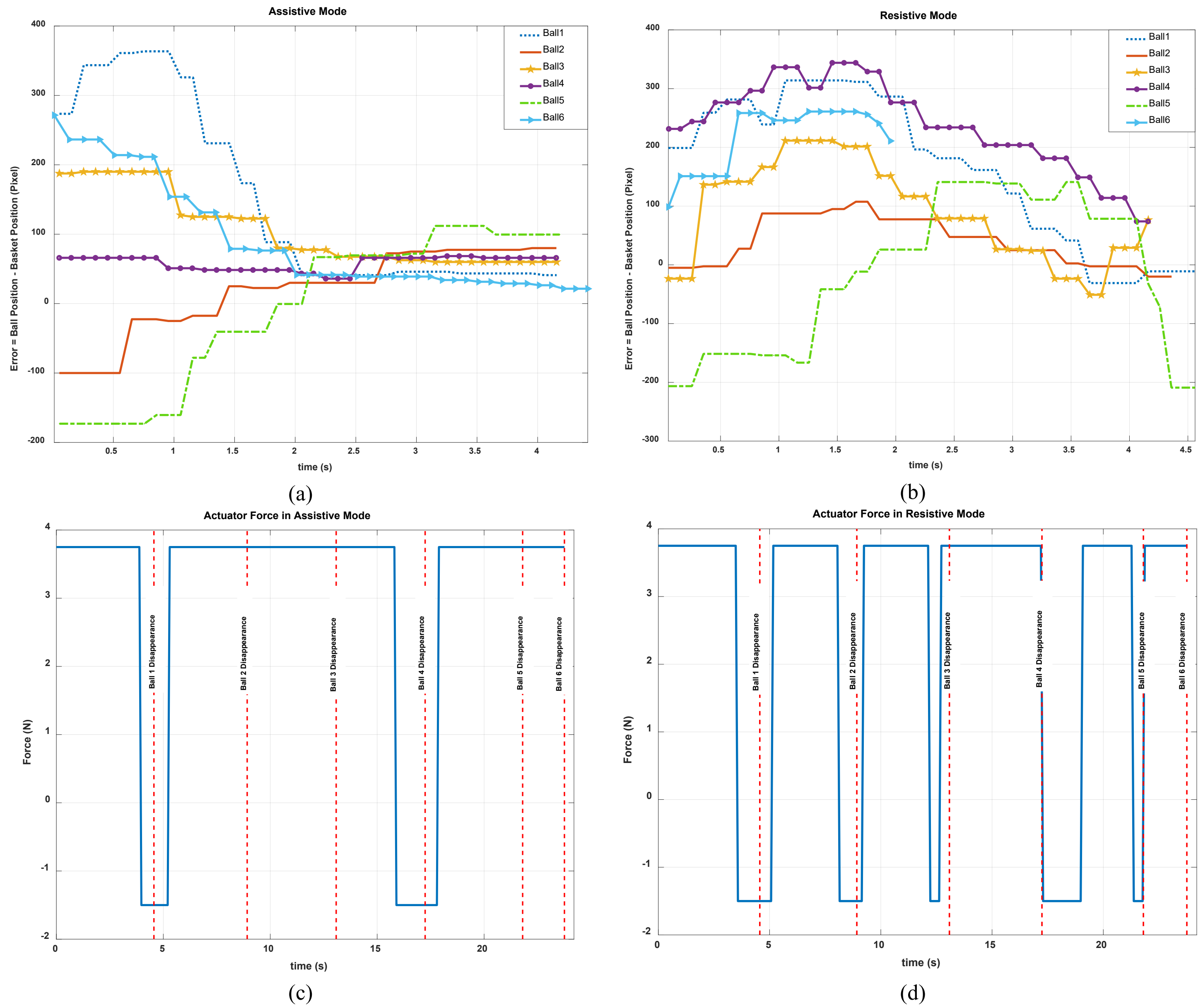


Fig. 5: Actuated game, (a) Position error in the assistive mode, (b) Position error in the resistive mode, (c) Actuator force in the assistive mode, (d) Actuator force in the resistive mode.

Table 1 shows information on the system's delay. Data transfer between the two Raspberry Pi boards requires 10 ms, while game screen updates occur every 50 ms. Command signals are also transmitted to the Arduino board within 0.1 ms.

The user can utilize the device for playing other games beyond the one provided. In that case, the device will function as a computer mouse, allowing the user to maneuver the cursor on the screen. As a result, the device is compatible with any commercial computer game that can be played using a computer mouse. It should be noted that in this case, the total time delay is 13 ms.

TABLE 1: Delay of the system.

	Delay (ms)	**Boards**	**Connection**
Data transfer	10	Raspberry Pi 4 – Raspberry Pi Pico	Wireless
Game events update	50	Raspberry Pi 4	-
Arduino serial communication	0.1	Raspberry Pi 4 – Arduino	USB
Moving mouse for other games	13	Raspberry Pi 4 – Raspberry Pi Pico	Wireless

5. Conclusion

The primary objective of this study was to develop a functional rehabilitation device that can be integrated with a computer game, enabling patients to play the game in real time during their rehabilitation sessions. The system employs a wireless connection between the device (equipped with Raspberry Pi Pico) and the computer (Raspberry Pi 4). The device is capable of transferring data with delay of 10 ms by incorporating multithreading and IRQ capabilities into the program. Given this minor delay, patients can play the game in real time with four different modes of playing including assistive and resistive modes. In addition to different game modes, the game's difficulty can be customized by altering the speed.

References

[1] Enas S. Lawrence, Catherine Coshall, Ruth Dundas, Judy Stewart, Anthony G. Rudd, Robin Howard, and Charles D. A. Wolfe, "Estimates of the prevalence of acute stroke impairments and disability in a multiethnic population," *Stroke*, vol. 32, no. 6, pp. 1279–1284, Jun 2001. https://doi.org/10.1161/01.STR.32.6.1279

[2] Kate E Laver, Belinda Lange, Stacey George, Judith E Deutsch, Gustavo Saposnik, and Maria Crotty, "Virtual reality for stroke rehabilitation," *Cochrane Database of Systematic Reviews,* no. 11, Nov 2017. https://doi.org/10.1002/14651858.CD008349.pub4

[3] Grigore Burdea, Nam Kim, Kevin Polistico, Ashwin Kadaru, Namrata Grampurohit, Jasdeep Hundal, and Simcha Pollack, "Robotic table and serious games for integrative rehabilitation in the early Poststroke phase: Two case reports," *JMIR Rehabilitation and Assistive Technologies*, vol. 9, no. 2, Jan 2022. https://doi.org/10.2196/26990

[4] Marek Sierotowicz, Nicola Lotti, Laura Nell, Francesco Missiroli, Ryan Alicea, Xiaohui Zhang, Michele Xiloyannis, Rüdiger Rupp, et al., "EMG-driven machine learning control of a soft glove for grasping assistance and rehabilitation," *IEEE Robotics and Automation Letters*. vol. 7, no. 2, pp. 1566-1573, Apr 2022. https://doi.org/10.1109/LRA.2021.3140055

[5] Gerdienke B. Prange-Lasonder, Bob Radder, Anke I. R. Kottink, Alejandro Melendez-Calderon, Jaap H. Buurke, and Johan S. Rietman, "Applying a soft-robotic glove as assistive device and training tool with games to support hand function after stroke: Preliminary results on feasibility and potential clinical impact," *International Conference on Rehabilitation Robotics (ICORR)*, pp. 1401-1406, Jul 2017. https://doi.org/10.1109/ICORR.2017.8009444.

[6] Sejal Ghate, Longteng Yu, Kang Du, Chwee T. Lim, and Joo C. Yeo, "Sensorized fabric glove as game controller for rehabilitation," *IEEE Sensors*, pp. 1-4, Oct 2020. https://doi.org/10.1109/SENSORS47125.2020.9278938.

[7] Rob Endure, "Game Latency Tolerance," Gamasutra, 2014. https://www.gamasutra.com/blogs/RobEndure/20141001/226370/Game_Latency_Tolerance.php.

[8] Witold Kinsner, *Microcontroller, Microprocessor and Microcomputer Interfacing for Real-Time Systems*. Winnipeg, MB: OCO Research, Sep 2020, 973 pages. {ISBN: 978-0-9939347-5-9, eBook}

[9] Sus Lundgren and Staffan Bjork, "Game mechanics: Describing computeraugmented games in terms of interaction," *In Proceedings of TIDSE*, vol. 3, Nov 2003.

[10] John K. Perng, Fisher Brian, Seth Hollar, and Kristofer S.J. Pister, "Acceleration sensing glove (ASG)," in *Proc. Third International Symposium on Wearable Computers*, pp. 178-180, Oct 1999. https://doi.org/10.1109/ISWC.1999.806717