Robotic Accessibility Assistant (RAA)

Tais Mota
Mechanical Engineering
University of South Florida
Tampa, United States
mota5@usf.edu

Ohad Nataf
Mechanical Engineering
University of South Florida
Tampa, United States
ohadnataf@usf.edu

Faith El Massari
Mechanical Enginneering
University of South Florida
Tampa, United States
faithe200@usf.edu

Morgan Walkinshaw
Mechanical Engineering
University of South Florida
Tampa, United States
mwalkinshaw@usf.edu

Redwan Alqasemi
Mechanical Engineering
University of South Florida
Tampa, United States
alqasemi@usf.edu

Rajiv V. Dubey
Mechanical Engineering
University of South Florida
Tampa, United States
dubey@usf.edu

***Abstract*— Individuals with muscular dystrophy and spinal cord injuries often experience upper-limb disabilities and rely on Power wheelchairs for their mobility. While these wheelchairs enhance their quality of life, they do not fully address challenges in activities of daily living (ADL). Specifically, the stiffening of tendons and ligaments in hand joints makes it difficult for individuals to perform accessibility tasks such as pressing elevator buttons and scanning access cards [1] . This paper presents an affordable, simple, and convenient solution to these challenges.**

The proposed robotic arm offers several unique features. Equipped with a 4-DOF system (3 revolute joints and 1 prismatic joint), it enables easy 3D navigation, controlled via an assistive joystick. Its electrical system uses a Raspberry Pi [2] as the primary microcontroller and an Arduino [3]for joystick input. The four interchangeable end-effectors include a card holder, a ballpoint for pushing buttons, a hook, and a 1 DOF gripper. Other features include 3D printed covers, easy assembly, and quick attachment/detachment. Additionally, this accessory meets National ADA Standards for accessible design [4], ensuring the wheelchair adheres to a width clearance of 32 inches.

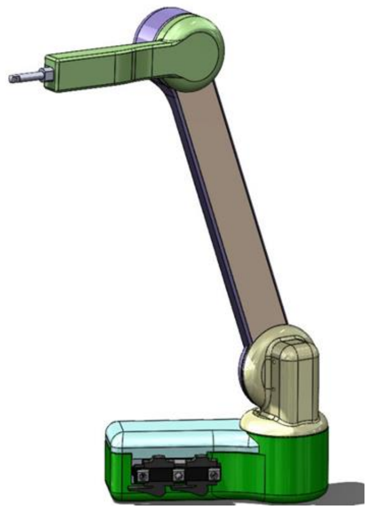
***Keywords*— ADL tasks, Robotic Arm, Power Wheelchair, Elevator Accessibility, Raspberry Pi, Arduino, Upper-Limb Disability, Assistive Technology.**

I. Introduction

According to the most recent United States Census, approximately 19% of the U.S. population, or about 56.7 million Americans, have a disability. Among these, approximately 3.6 million use wheelchairs for mobility assistance [5]. Muscular dystrophy and spinal cord injuries are common conditions among wheelchair users, often necessitating the use of electric wheelchairs due to the increased constriction and stiffening of tendons and ligaments [6]. This reduced arm strength makes it difficult to manually operate a wheelchair, and tasks requiring users to extend their arm are especially challenging.

There has been minimal research exploring the technologies that enhance the experience of power wheelchair users. Some studies focus on improved navigation, such as implementing vibrotactile haptics control systems for wheelchair users.[7] Others have attempted to integrate robotic arm accessories to assist with ADL. These solutions, however,, are often expensive, bulky, and require lengthy adaptation processes.[8] Moreover, there is a significant lack of research addressing the tasks that the proposed robotic arm will target: pressing elevator buttons and scanning cards.

This paper aims to present an affordable, simple, and convenient robotic arm accessory for power wheelchairs that addresses the specific tasks of pushing elevator buttons and scanning cards. The robotic arm features two main modules: the first module includes the robotic arm main frame and electrical components, while the second module includes the LCD screen, joystick, and Arduino Nano. The following sections will detail the design, manufacturing, evaluation, and optimization of the model. Figure 1 demonstrates the 3D model assembly of the robotic arm.



.

Fig. 1. *3D model of robotic arm*

II. Design Implementation

A. Measurements

Prior to acquiring components, measurements were made to estimate an adequate length of the robotic arm and determine the minimum torque required for each jointThis ensures that the robotic arm can accurately reach and press elevator buttons in any building.

1) Arm Length

To facilitate the study, a reference coordinate system was developed, as depicted in Figure 2. This system considers a user sitting in the wheelchair, with the reference point at the location of the wheelchair's joystick. The x-axis corresponds to right and left distances, the y-axis to front and back distances, and the z-axis to up and down distances.

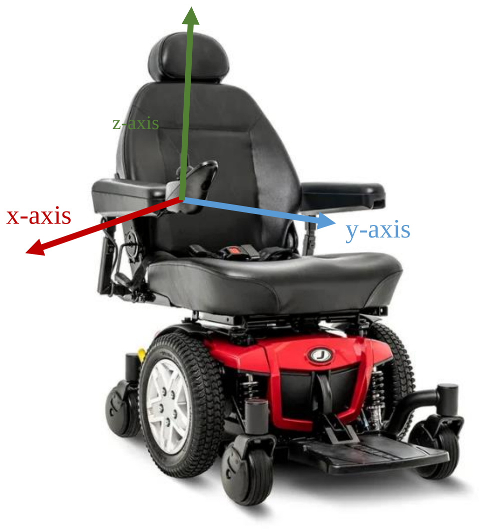


Fig. 2. *Reference coordinate system*

Secondly, to ensure universal compatibility with elevator buttons, the button panel dimensions were carefully considered. Buttons panel have an standard height of 8 inches, a width of approximately 6 inches, and it is centered 42 inches from the floor. Door scanners are located at a height of 34 inches up to 48 inches from the floor These measurements provide essential parameters for positioning and reach considerations regarding the robotic arm's functionality, ensuring that the arm can effectively interact with elevator buttons across a range of heights and widths.

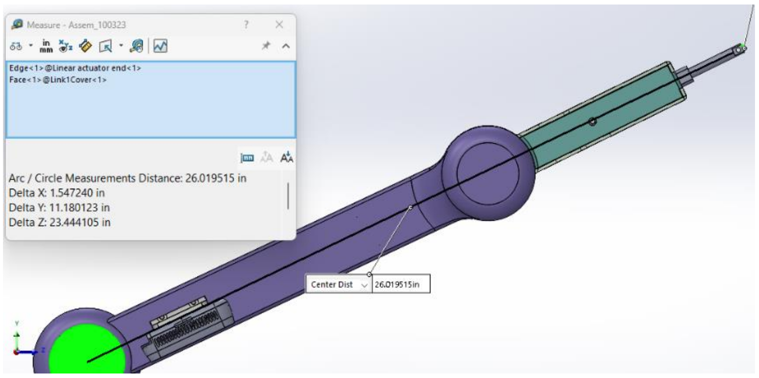
Table I. *Arm measurements related to door scanner distances*

Arm Measurements (Door Scanner)		
45° to door scanner	x-axis: 12 in y-axis: 0 in z-axis:15.5 in Resultant: 19.602 in	
90° to door scanner	x-axis: 14 in y-axis: 0 in z: 15.5 in Resultant: 20.886 in	x-axis: 12 in y-axis: 0 in z: 15.5 in Resultant: 19.602 in

Table II. *Arm measurements related to elevator buttons*

Arm Length Measurements (Elevator Buttons)		
90° inside elevator	***Distance to bottom left button***	***Distance to top right button***
	x-axis: 3 in y-axis: 12 in z: 18 in Resultant: 21.703 in	x-axis: 3 in y-axis: 15 in z: 18 in Resultant: 23.622 in
Backing into elevator at an angle	***Distance to bottom left button***	***Distance to top right button***
	x-axis: 5in y-axis: 15in z-axis: 4.5in Resultant: **16.439** in	x-axis: 11in y-axis: 15in z-axis: 18 in Resultant: **25.884** in
45° inside elevator	***Distance to center of buttons***	***Distance to far most button***
	x-axis: 9 in y-axis: 11 in z-axis:18 in Resultant: 22.935 in	x-axis: 12 in y-axis: 11 in z-axis: 18 in Resultant: 24.269 in

Table 1 and Table 2 present comprehensive measurements considered when determining the robotic arm's length. The maximum distance observed in the measurements is 22.4 inches. Based on this observation and the availability of parts, a decision was made to set the arm's overall length at approximately 26.4 inches or 671 mm, as depicted in Figure 3. The first arm length is a total of 18 inches, second arm length is 8 inches, and the linear actuator [9] length is 2 inches at its maximum position. Additional length was incorporated to accommodate the varying heights of power wheelchairs and to provide flexibility in arm positioning based on user preference.



Edge<1>@Linear actuator end<1>
Face<1>@Link1Cover<1>

Arc / Circle Measurements Distance: 26.019515 in
Delta X: 1.547240 in
Delta Y: 11.180123 in
Delta Z: 23.444105 in

26.019515in

Fig. 3. *Depicts the overall length of the robotic arm.*

B. Force Calculations

Upon selecting the overall length of the robotic arm, force calculations were conducted to determine the required torque for each joint motor and to determine a suitable basis for the material and shape for the main body frame. Calculations were conducted based on two main factors: Length and linear actuator force. The length of each arm is 18 inches, 8 inches and 2 in respectively from down up. The standard force to push an elevator button is 1lb or 4.5 N and to press buttons to open automatic doors is 3lbs or 13.5 N. Finally, two main scenarios were considered during the calculations, when the arm is upright and completely horizontal. By considering the two scenarios, the maximum forces can be calculated as depicted in Table 3.

Table III: *Depicts the maximum torques on each joint of the robotic arm*

Torque Calculations		
Vertically Downward (Maximum torque position 1)	***Torque Joint 1***	1.6155 N-m
	Torque Joint 2	3.6729 N-m
Horizontally (Maximum torque position 2)	***Torque Joint 1***	5.0718 N-m
	Torque Joint 2	12.014 N-m
	Torque Joint 3	11.019 N-m

These measurements play a pivotal role in shaping the arm's design, ensuring seamless compatibility with elevator buttons and door scanners while also guaranteeing optimal reach and comfort. Vertical downward torque is meticulously calculated at 4.5 Newton, while horizontal torque, specifically for activating automatic door buttons, is determined to be 13.5. Notably, the calculation process for horizontal torque is considerably more complex. It factors in various elements, including the deployment force of the linear actuator, the weight of rods and links, and the combined weights of components and motors. These comprehensive calculations form the cornerstone of the subsequent components overview section, providing a detailed blueprint for the arm's construction and functionality.

C. Components Overview

Once the calculations and measurements were complete, the main components of the robotic arm were selected and organized into a flowchart, as depicted in Figure 4.

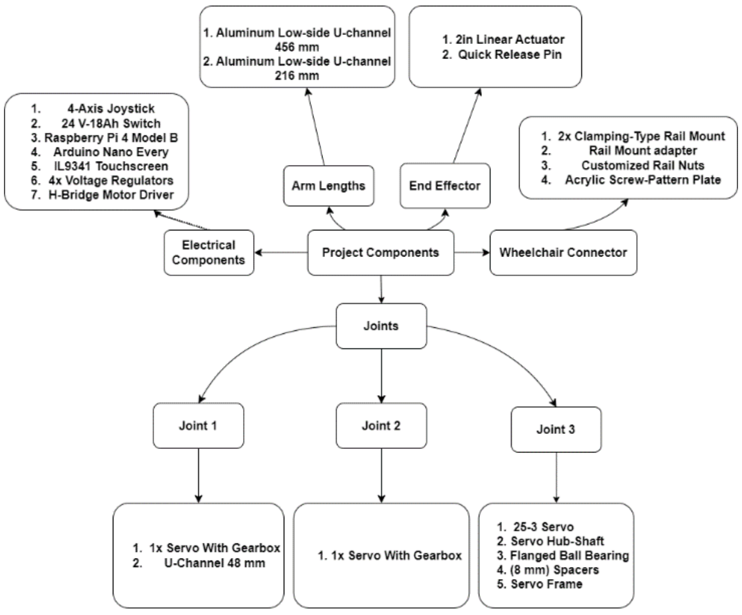


Fig. 4. *Flowchart depicting all the main components of the robotic arm with their respective locations and roles.*

For the arm lengths, aluminum low-side U-channels were selected, as they provide three main features: lightweight yet strong properties, patterned screw holes for easy part attachment, and a u-shape that prevents torsional stress.

To accommodate the forces depicted in the calculations, two gearbox servo motors (1.53 sec/60°, 6.7 RPM, 3150 oz-in torque, 200° rotation) [10] were selected for joints one and two. These motors allow for precise movement of the robotic arm while remaining within the force parameters.

The third joint possesses a 35kg, 360-degree, which corresponds to the same servo as the one used for joints 1 and 2 in terms of torque, the only difference is the range of 360 degrees. It allows for complete retractability of the robotic arm while adhering to force calculation measurements. The servo hub shaft, flanged ball bearing, spacers and servo frame are used to ensure that no load is rested on the servo motor, as it is only meant to enable rotation.

The quick-connect mechanism, electrical and end-effector components will be detailed in the hardware implementation section.

III. Hardware Implementation

A. Robotic Arm Enclosures

A set of enclosures were meticulously designed, and 3D printed with PLA plastic to properly encase the robotic arm. These enclosures feature a user-friendly quick-connect design, ensuring easy assembly. They serve to shield all electronic components, safeguarding the arm against dust and external impacts without restricting motion. If a user would like to remove or exchange any of the enclosure, they can simply pull the enclosure from its current location without the need to remove or screws of ties. This is possible due to two main quick-connect mechanism implemented, as depicted in Figure 5. Additionally, these measurements ensure the arm will be able to comfortably reach. Figure 6 showcases the 3-D printed enclosures in contrast to non-enclosed electrical components for reference.

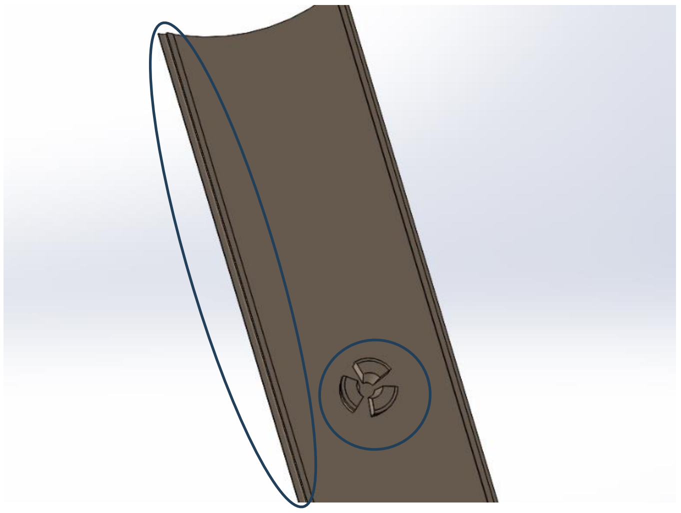


Fig. 5. *Showcases the two quick-connect mechanisms implemented in the robotic arm enclousures*

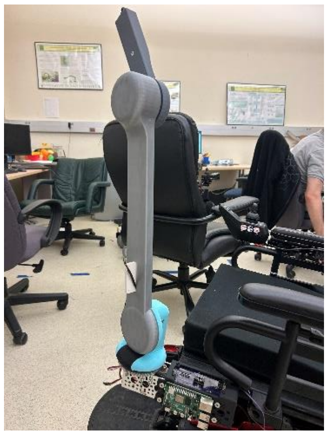


Fig. 6. *Partial robotic arm assembly without electrical components enclosures for reference*

B. *Wheelchair Quick-Connection*

A main feature of Robotic Elevator Assistant is to highlight the RAA's ease with quick assembly and connection to the wheelchair, a simple quick-connect system was developed by employing the standardized power wheelchair rai. The system composed of the following components: quick-release clamps (Fig. 7) [11], a rail slot mount adaptor (Fig. 8) [12], a customized part made of architectural 6063 aluminum bar with a centered M5 threaded hole, 1'' high, 1/4'' thick x 1-1/2'' wide (Fig. 9), and an acrylic plate designed to adapt the quick-release clamp screw pattern to the screw pattern of the internal aluminum C-brackets (Fig. 10).

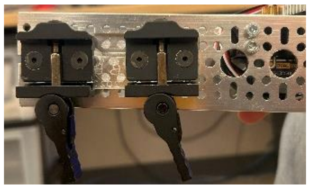


Fig. 7. *Quick release clamps mounted on main frame.*

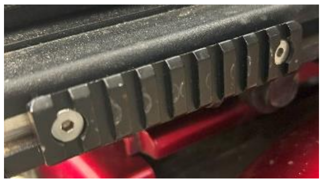


Fig. 8. *Rail slot mount adaptor screwed into customized rail nuts.*

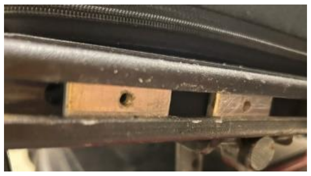


Fig. 9. *Rectangular-shaped rail nuts with centered threaded hole.*

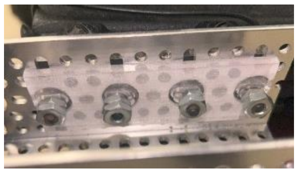


Fig. 10. *Acrylic plate installed with respective wheelchair rail.*

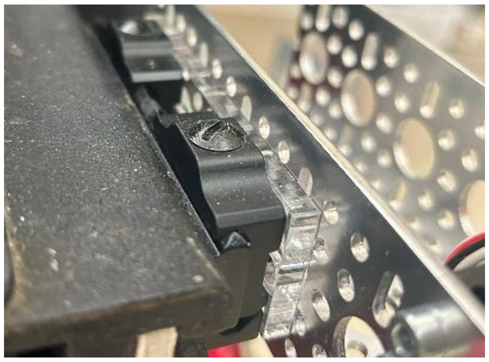


Fig. 11. *Depicts the final quick-connect system while attached to wheelchair and robotic arm.*

C. *Printed Circuit Boards*

Two printed circuit boards (PCBs) [13] were designed using the EasyEDA [14] software to enhance wiring efficiency, conserve space and optimize electrical connections. Both and PCB2 are featured below in schematic diagrams.

1) *PCB1-V2 – Module 1*

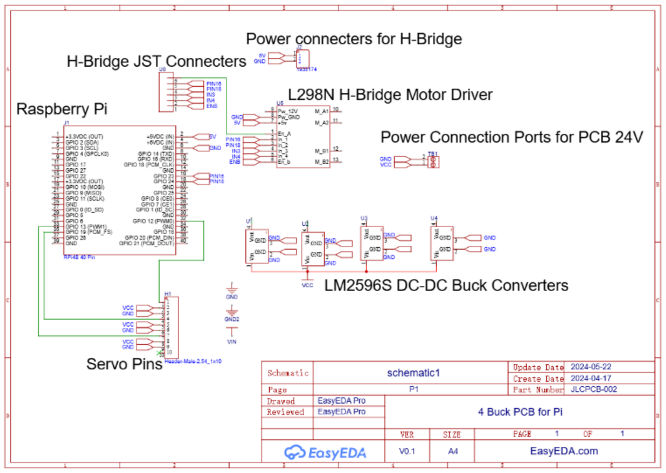


Fig. 12. *Module 1 (Raspberry Pi, voltage regulators, motors and motor driver) schematic.*

The development and implementation of PCB1 Version 2 (PCB1-V2) have been integral to the functionality of the robotic arm designed for pressing elevator buttons and scanning cards. Featuring the Raspberry Pi 4B, four adjustable voltage regulators [15], and an H-bridge for motor control [16], PCB1-V2 effectively utilizes the wheelchair's 24V battery. The voltage regulators ensure that each electrical component receives the appropriate current and voltage. Specifically, the regulators, which can output a maximum current of 3A and a voltage range of 1.5V to 35V, are crucial for maintaining the stability and performance of the robotic arm. For instance, the Raspberry Pi, requiring 3A at 5V, is connected to the 5V port of the H-bridge, which itself is linked to a voltage regulator.

Each servo motor, operating at 7.4V with a maximum current draw of 3A, is similarly connected to a dedicated voltage regulator. This configuration ensures that the servos receive consistent power, crucial for the precise operation of the robotic arm. Additionally, the connectors depicted in Figure 12 are used for the H-bridge, servo pins, and power supply, with a deliberate design choice to not utilize all H-bridge connections. This allows the PCB to be positioned perpendicular to the H-bridge, optimizing space, accessibility, and wiring efficiency.

The successful implementation of PCB1-V2 facilitates easy and reliable connections for all electrical components, minimizing wiring and enhancing the overall functionality of the robotic arm. This design ensures that individuals with motor disabilities can independently navigate spaces by efficiently pressing elevator buttons and scanning cards, thus significantly improving their autonomy and access to various environments.

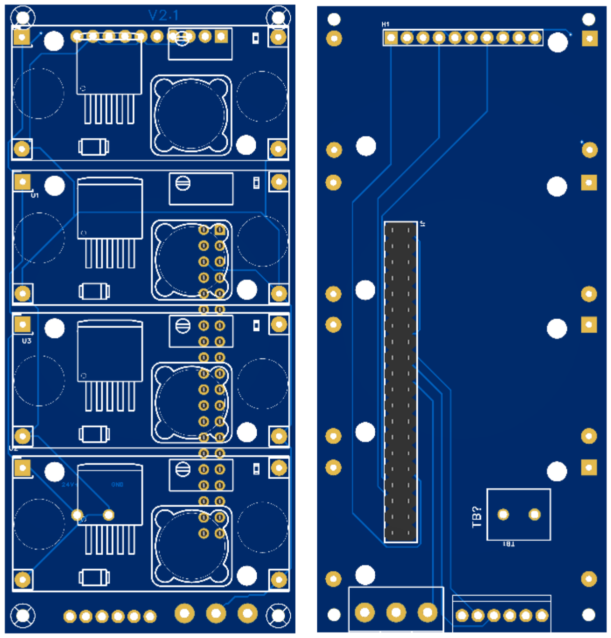


Fig. 13. *PCB1-V2 top view and PCB1-V2 bottom view, respectively.*

2) *PCB2 – Module 2*

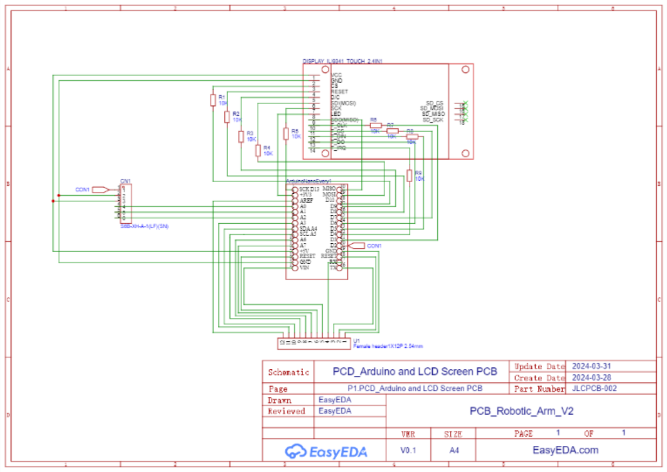


Fig. 14. *Module 2 (LCD screen, joystick and Arduino) schematic.*

Figure 14 is a schematic featuring the Arduino Nano Every, LCD Screen (Display ILI9341 Touch) [17], one Sullins 12-pin standard male connector, and one standard JST 6-pin connector heads. These connections are established according to the power and signal requirements specified by the data sheet of each electronic component. Additionally, the 6-pin connector head (CN1) serves to firmly connect the joystick wiring to the Arduino board while the 12-pin connector (U1) provides easy access to all unused Arduino pins. Figures 15 and 16 display the back view and side view of the corresponding PCB2, respectively.

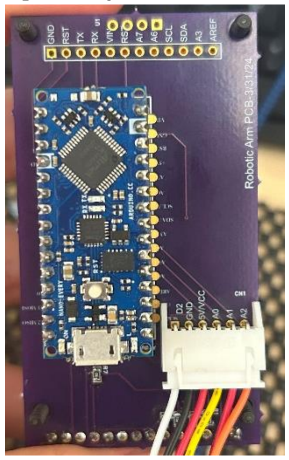


Fig. 15. *Back-View of PCB2 with all integrated components*

Fig. 16. *Side-View of PCB board depicting the LCD screen and its connection to PCB2*

D. *End-Effectors*

The designs discussed here include two variations of the end-effector attachment options for the robotic arm. The first end-effector is a combined card slot and button presser. This design was prototyped using PLA 3D-printing and is part of the button presser/card scanner subassembly [18] . The 3D-printed part attaches to the end of the linear actuator using an 18-8 Stainless Steel Ring-Grip Quick-Release Pin [19]. One side of the end-effector is shaped to fit the linear actuator, while the other side features a circular clip designed to hold cards or similar items. This design ensures ease of access for the user and provides a large surface area for inserting or removing cards.

On the button presser end, the top surface has a rounded rubber tip, which enhances the ease of pressing elevator buttons when the actuator is activated (Fig. 17). This combination of features makes the end-effector both functional and user-friendly, allowing individuals to independently navigate spaces by efficiently using the robotic arm for pressing buttons and scanning cards.

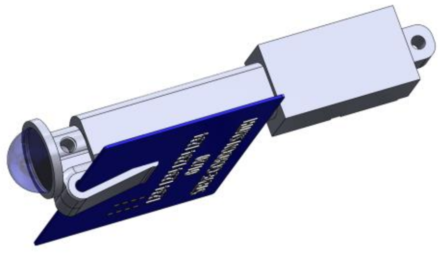


Fig. 17. *Depicts the button presser and card scanning end-effector placed on the actuator end,*

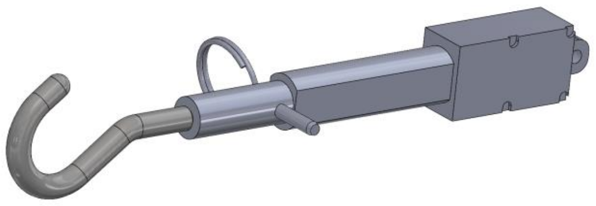


Fig. 18. *Depicts the hook end effector, attached via the actuaor end and secured witht the quick-release pin.*

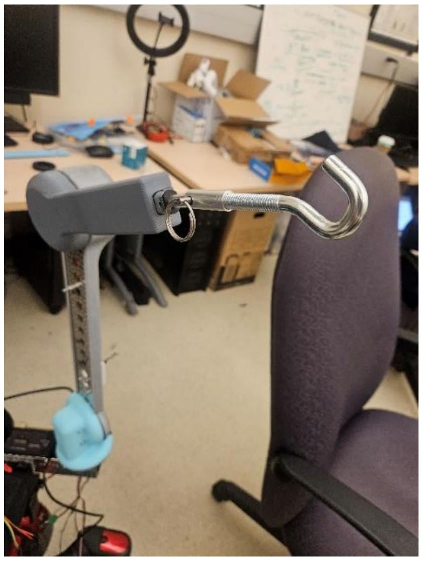


Fig. 19. *Shows the hook end-effector attached to the robotic arm prototype.*

Figures 18 and 19 illustrate the second end-effector designed for the Robotic Accessibility Assistant (RAA). This end-effector extends the primary function of pressing elevator buttons by incorporating a hook. The hook enables users to pick up objects that might have fallen to the floor, addressing another challenge faced by individuals who use power wheelchairs. This dual functionality significantly enhances the usability and convenience of the RAA, providing a practical solution for everyday tasks beyond just navigating and accessing buildings

E. Module 2 – Enclosure

The Module 2 enclosure (Fig.20) houses two main assemblies necessary for allowing user interaction with the robotic arm: the joystick for operating the arm and the Module 2 assembly, which includes the PCB2, Arduino Nano arm and an LCD display. The joystick is connected to the PCB through wires, while the Arduino and LCD display are pinned directly to the PCB. Communication with the arm is achieved by a wire connecting the Arduino to the Raspberry Pi on the other side of the wheelchair. The enclosure is a critical component to the operation of the project, and its design must include considerations such as structural integrity, sleekness, and prudence towards disabled users.

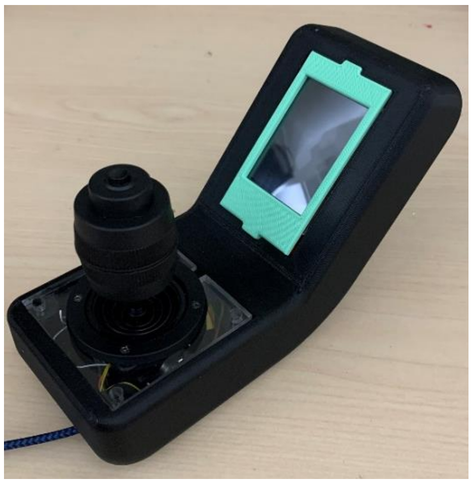


Fig. 20. *Final model of Module 2 including all its hardware components*

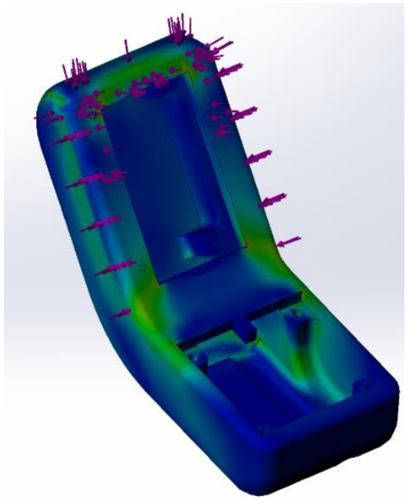


Fig. 21. *Module 2 enclosure in response to FEA analysis.*

As part of the design for user interface process, plausible conditions for applied loads on the chassis were discussed, and modifications were implemented in response to a Finite Element Analysis (FEA). In order of most to least critical, two common scenarios were the full weight of a human hand on the face of where the LCD will be accessed (Fig. 21), and on the joystick mounting surfaces. Potential failure points were realized at the point of curvature in the chassis, and where the joystick mounting surfaces connect with the chassis. The response was to add material at these locations and to introduce larger radius to corner transitions to reduce stress concentrations.

IV. Software Implementation

Before proceeding with the software implementation, several critical hardware factors must be considered to ensure optimal performance and compatibility. Firstly, to provide sufficient processing power for the anticipated automation of the robotic arm, a Raspberry Pi 4 Model B was selected as the primary processing unit. This model offers a robust platform with ample computational capabilities and flexibility for handling complex tasks and future expansions.

Secondly, a pivotal consideration is the analog nature of the 4-axis joystick, which relies on potentiometers to transmit data. The challenge here is that the Raspberry Pi can only process digital inputs. To bridge this gap, an intermediary device is required. The Arduino Nano serves this crucial role, utilizing its analog pins to read the joystick's analog signals. This choice allows for precise retrieval and organization of the joystick data before it is passed on to the Raspberry Pi.

The communication between the Raspberry Pi and the Arduino Nano is established via I2C serial communication [20], a protocol well-suited for microcontroller interfacing due to its simplicity and efficiency. This communication is facilitated by a USB cable connecting the two microcontrollers, ensuring a reliable and straightforward data transfer method.

Additionally, the Arduino Nano was chosen because of its compatibility with the ILI9341 LCD touchscreen implemented in the robotic arm. This compatibility simplifies the integration process and allows for a seamless user interface. The touchscreen provides users with an intuitive control panel, enhancing the overall functionality and user experience of the robotic arm system.

Moreover, other hardware factors, such as power supply requirements, physical layout, and mounting solutions, were carefully considered to ensure the system's reliability and ease of use. The choice of components and their integration into the system were guided by the need for durability, ease of maintenance, and the ability to withstand the operational environment of the robotic arm.

Finally, the Raspberry Pi and Arduino units were divided into modules one and two respectively (Fig.21); these will be discussed below.

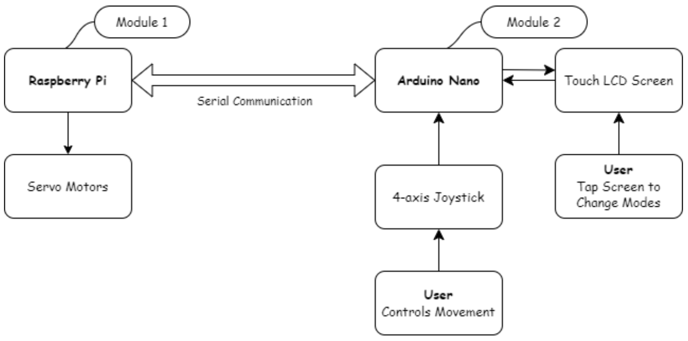


Fig. 22. *Flowchart depicting the software signaling between hardware components.*

A. *Module 1 – Software Implementation*

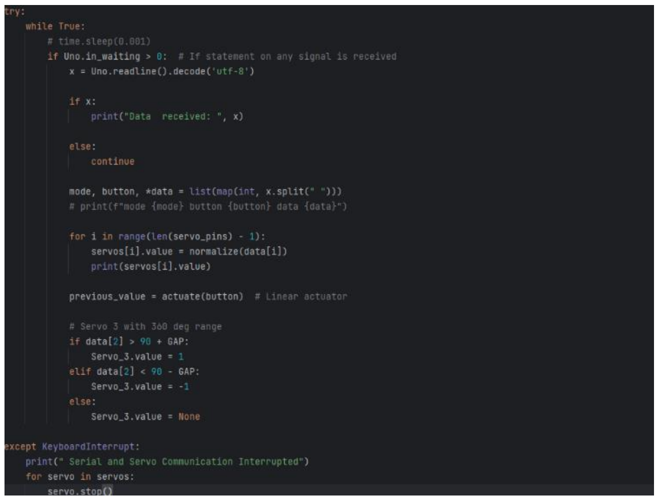
The primary function of the Module 1 software implementation is to receive data from Module 2, decode it, and accurately control the robotic arm's servo motors. However, to achieve precise control, it is essential to address the issue of servo jittering caused by the default Raspberry Pi GPIO drivers. The default Raspberry Pi *GPIO drivers* cause [21] servo jittering due to servo and the pi PWM incompatibility [22]. To address this issue, the *gpiozero* library [23] was implemented. This library uses the *pigpio* library driver [24] and its DMA sampling for [25] a more precise edge timing, increasing sampling rates and eliminating servo jitters. Furthermore, its interface facilitates accurate control over each servo by offering parameters for initial position angles, maximum and minimum angles, and by setting pre-defined class functions for seamless servo movement to any desired position or direction. Additionally, the Raspberry Pi in-built *serial* library was used to allow for I2C communication with the Arduino Nano.

The software script begins by importing all necessary libraries, including the *serial* library for serial communication with the Arduino and the *gpiozero* library for controlling the servo motors and the linear actuator. The script initiates serial communication with the Arduino using the *serial* library. Servo motors are then instantiated using the *gpiozero* library's built-in *Angular Servo* class, while the linear actuator is instantiated using the *Motor* class.

To manage the data received from the Arduino, the script includes a function to normalize this data. Another function is developed to control the linear actuator motor, defining a range of values that correspond to the forward or backward movement of the servos.

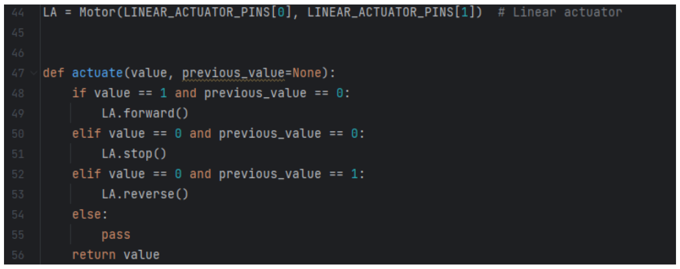
The main loop of the script continuously reads serial data from the Arduino (Fig. 23). This data is averaged and processed to generate commands for the servo motors, ensuring they move according to the received commands.

This structure ensures efficient and responsive control of the robotic arm's components, integrating data normalization and motor control into a seamless operation cycle.



```python
try:
    while True:
        # time.sleep(0.001)
        if Uno.in_waiting > 0:  # If statement on any signal is received
            x = Uno.readline().decode('utf-8')

            if x:
                print("Data  received: ", x)

            else:
                continue

            mode, button, *data = list(map(int, x.split(" ")))
            # print(f"mode {mode} button {button} data {data}")

            for i in range(len(servo_pins) - 1):
                servos[i].value = normalize(data[i])
                print(servos[i].value)

            previous_value = actuate(button)  # Linear actuator

            # Servo 3 with 360 deg range
            if data[2] > 90 + GAP:
                Servo_3.value = 1
            elif data[2] < 90 - GAP:
                Servo_3.value = -1
            else:
                Servo_3.value = None

except KeyboardInterrupt:
    print(" Serial and Servo Communication Interrupted")
    for servo in servos:
        servo.stop()
```

Fig. 23. *Raspberry Pi script main loop*



```python
LA = Motor(LINEAR_ACTUATOR_PINS[0], LINEAR_ACTUATOR_PINS[1])  # Linear actuator


def actuate(value, previous_value=None):
    if value == 1 and previous_value == 0:
        LA.forward()
    elif value == 0 and previous_value == 0:
        LA.stop()
    elif value == 0 and previous_value == 1:
        LA.reverse()
    else:
        pass
    return value
```

Fig. 24. *Linear actuator main controlling function*

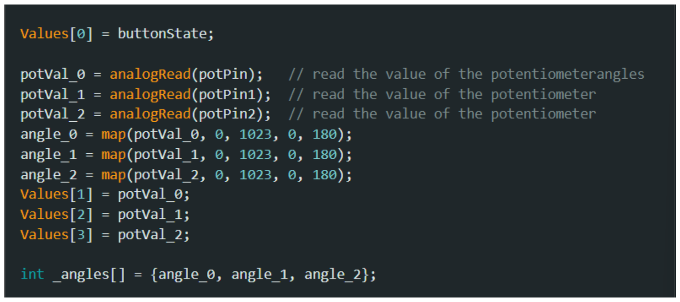
The linear actuator functionality is depicted in Figure 24, showing the operation of the *actuate* function. This function manages the linear actuator using the joystick's pushbutton commands. When the button is pressed once, the actuator moves forward. Pressing the button again stops the actuator and pressing it a third time reverses its direction. This sequence allows for precise control over the actuator's movement, providing a user-friendly interface for adjusting the linear actuator motion.

B. *Module 2 – Software Implementation*

The main goal of the Module 2 software implementation is to develop a C++ based script to initialize and control the touch LCD screen and to retrieve analog data from the joystick controller. This script is responsible for setting up the LCD screen, receiving data from the joystick, organizing this data, and then transmitting it to the main microcontroller unit, the Raspberry Pi, via serial communication. By consolidating these tasks, the Module 2 software ensures seamless data handling and communication with the Raspberry Pi for efficient control of the robotic arm.

To accomplish these tasks, the code utilizes the *Adafruit_ILI9341.h* [26], *Adafruit_GFX.h* [27] and *URTouch.h* [28] libraries. The Adafruit GFX library collaborates with *Adafruit_ILI9341* to configure the graphics of the LCD, while the *URTouch* library enables the touchscreen functionality of the LCD screen.

The Arduino inputs the joystick's data using its in-built *analogRead* function. Then, it maps these values between 0 and 1023 received from the joystick into 0 and 180, which are more convenient when dealing with servos angles. (Fig. 25).



```
Values[0] = buttonState;

potVal_0 = analogRead(potPin);   // read the value of the potentiometerangles
potVal_1 = analogRead(potPin1);  // read the value of the potentiometer
potVal_2 = analogRead(potPin2);  // read the value of the potentiometer
angle_0 = map(potVal_0, 0, 1023, 0, 180);
angle_1 = map(potVal_1, 0, 1023, 0, 180);
angle_2 = map(potVal_2, 0, 1023, 0, 180);
Values[1] = potVal_0;
Values[2] = potVal_1;
Values[3] = potVal_2;

int _angles[] = {angle_0, angle_1, angle_2};
```

Fig. 25. *Displays the Functions used to receive and organize the data received from the joystick.*

The final function of the Arduino is to send these values to the Raspberry Pi. The angles, the mode number and joystick button values are ordered and sent to the Raspberry Pi 4 using the Arduino in-built *Serial.println* function.

V. Conclusion and Future Work

This paper details the design and implementation of a cost-effective, user-friendly robotic arm accessory aimed at assisting individuals with upper-limb disabilities in performing accessibility tasks such as pressing elevator buttons and scanning access cards.

This paper details the design and implementation of a cost-effective, user-friendly robotic arm accessory aimed at assisting individuals with upper-limb disabilities in performing accessibility tasks such as pressing elevator buttons and scanning access cards. The robotic arm integrates seamlessly with power wheelchairs, enabling users to independently navigate spaces and access restricted areas without assistance. Featuring a 4-DOF system with three revolute joints and one prismatic joint, the arm offers flexible 3D navigation and is controlled via an assistive joystick, powered by a Raspberry Pi and Arduino for precise and reliable operation. The design emphasizes affordability, simplicity, and ease of assembly, making it accessible to a broad user base. The inclusion of interchangeable end-effectors like a cardholder, button presser, hook, and 1 DOF gripper enhances its versatility in daily tasks.

Future work within the current scope of the project will involve quantitative and qualitative analysis to evaluate the arm's effectiveness in terms of user-friendliness, price, convenience, and universal compatibility.

Given ongoing advancements in technology, expanding the project's scope to include features of an automated system is a promising direction. This includes voice-activated navigation, artificial intelligence, and point-cloud camera systems for object detection, such as elevator buttons. Possibly integrating an automated system with the current manual system implies creating touchscreen commands to the LCD screen to toggle between these two modes. Additionally, the design and integration of more end-effectors such as grippers or door openers are a great alternative to expand the project.

By addressing these areas, the goal is to continually refine the robotic arm to better meet the needs of individuals with upper-limb disabilities, enhancing their independence and quality of life

Acknowledgment

The authors would like to thank the Florida Department of Education - Division of Vocational Rehabilitation for their support.

VI. References

[1] B. K. Birnkrant DJ, Bann CM, Apkon SD, Blackwell A, Brumbaugh D, Case LE, Clemens PR, Hadjiyannakis S, Pandya S, Street N, Tomezsko J, Wagner KR, Ward LM, Weber DR & DMD Care Considerations Working Group " Diagnosis and management of Duchenne muscular dystrophy " *The Lancet. Neurology*, vol. 17(3), 251–267, part 1: diagnosis, and neuromuscular, rehabilitation, endocrine, and gastrointestinal and nutritional management., 2018 Art no. 251-267., doi: https://doi.org/10.1016/s1474-4422(18)30024-3.

[2] *Raspberry Pi 4 Microcontroller*.

[3] (2023). *Arduino Nano*. [Online]. Available: Arduino.

[4] *Excerpt from 28 CFR Part36: ADA Standards for Accessible Design*. [Online] Available: https://www.floridabuilding.org/fbc/committees/accessibility/access_tac/adaag_0302.pdf

[5] (2013). *U.S. Disability Statistics : Facts and Figures* [Online] Available: www.disabled-world.com/disability/statistics/info.php

[6] E. M. Yiu and A. J. Kornberg, "Duchenne muscular dystrophy," *Journal of Paediatrics and Child Health*, vol. 51, no. 8, pp. 759-764, 2015, doi: https://doi.org/10.1111/jpc.12868.

[7] L. Devigne *et al.*, "Power Wheelchair Navigation Assistance Using Wearable Vibrotactile Haptics," *IEEE Transactions on Haptics*, vol. 13, no. 1, pp. 52-58, 2020, doi: 10.1109/TOH.2019.2963831.

[8] M. Chi, Y. Yao, Y. Liu, and M. Zhong, "Recent Advances on Human-Robot Interface of Wheelchair-Mounted Robotic Arm," *Recent Patents on Mechanical Engineering*, vol. 12, no. 1, pp. 45-54, 2019, doi: 10.2174/2212797612666190115151306.

[9] D. H. STORE, "DC HOUSE Mini Electric Linear Actuator Stroke 2"," January.

[10] Servocity. "Stingray-9 Servo Gearbox (1.53 sec/60°, 6.7 RPM, 3150 oz-in Torque, 200° Rotation)." Servocity. https://www.gobilda.com/stingray-9-servo-gearbox-1-53-sec-60-6-7-rpm-3150-oz-in-torque-200-rotation/ (accessed.

[11] (2023). *Tactical DBAL-A2 QD Quick Release/Detach Mount, Full Metal Optics Laser Sight DBAL A2 Mount Adapter Accessories for 20mm Picatinny Rail*.

[12] (2023). *GOTICAL Multi Slots Mlok to Picatinny Rail Adapter | Set of 9 slot Ar Picatinny Rail System*.

[13] J. LaDou, "Printed circuit board industry," *International Journal of Hygiene and Environmental Health*, vol. 209, no. 3, pp. 211-219, 2006/05/16/ 2006, doi: https://doi.org/10.1016/j.ijheh.2006.02.001.

[14] E. Company. "An Easier and Powerful Online PCB Design Tool." https://easyeda.com/ (accessed January, 2024).

[15] *WWZMDiB LM2596 Voltage Regulator DC to DC Converter 3.2-35V to 1.25-30V Buck Converter (5Pcs)*.

[16] AITRIP. " PACK L298N Motor Drive Controller Board DC Dual H-Bridge Robot Stepper Motor Control and Drives Module for Arduino Smart Car Power Compatible with Arduino UNO MEGA R3 Mega2560." https://www.amazon.com/Controller-H-Bridge-Stepper-Control-Mega2560/dp/B07WS89781/ref=sr_1_6?keywords=h-bridge&qid=1690903280&sprefix=H-Bridg%2Caps%2C124&sr=8-6&th=1 (accessed January, 2024).

[17] (2023). *HiLetgo 2.4" SPI TFT LCD Display 2.4 Inch ILI9341 Touch Panel LCD ILI9341 240x320 5V 3.3V*.

[18] C. V., "All You Need to Know About PLA for 3D Printing," 2023. [Online]. Available: https://www.3dnatives.com/en/pla-3d-printing-guide-190820194/.

[19] M. CARR. "18-8 Stainless Steel Ring." https://www.mcmaster.com/98404A101/ (accessed.

[20] R. Krauss, "Real-Time Python: Recent Advances in the Raspberry Pi Plus Arduino Real-Time Control Approach," 2020.

[21] W. W.Gay, *Raspberry Pi Hardware Reference*, 1 ed.: Apress Berkeley, CA, 2014, p. 248. [Online]. Available: https://www.researchgate.net/publication/316807063_Raspberry_Pi_Hardware_Reference.

[22] M. Bolic, "Chapter 3 - Electronics, Pervasive Cardiovascular and Respiratory Monitoring Devices," *Pervasive Cardiovascular and Respiratory Monitoring Devices*, no. Pages 73-123, 2023, doi: https://doi.org/10.1016/B978-0-12-820947-9.00011-8.

[23] B. Nuttall. GPIO Zero library Documentation [Online] Available: https://gpiozero.readthedocs.io/en/latest/index.html

[24] R. Hirst. Pigpio Library Documentation [Online] Available: https://abyz.me.uk/rpi/pigpio/index.html

[25] G. Kiarie, C. Wa Maina, and K. Nyachionjeka, "A low‐cost Raspberry Pi based time domain reflectometer for fault detection in electric fences," *IET Science, Measurement & Technology*, 2024, doi: 10.1049/smt2.12183.

[26] "Adafruit ILI9341 Arduino Library." https://github.com/adafruit/Adafruit_ILI9341?tab=readme-ov-file (accessed.

[27] P. Burguess. "Adafruit GFX Graphics Library." Adafruit. https://learn.adafruit.com/adafruit-gfx-graphics-library (accessed 2024).

[28] H. Karlsen. "Library: URTouch." Rinky-Dink Electronics. http://www.rinkydinkelectronics.com/library.php?id=92 (accessed 2024).