Robotic Mobility Assistive Device

A Design Project Report
Presented to the School of Electrical and Computer Engineering of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering, Electrical and Computer Engineering

Submitted By: Zhihao Xu, Yiqi Sun
MEngField Advisor: Hunter Adams
Degree Date: May 2024

Abstract

Master of Engineering Program
School of Electrical and Computer Engineering
Cornell University
Design Project Report

Project Title: Robotic Mobility Assistive Device

Author: Zhihao Xu, Yiqi Sun

Abstract: The Revival of Motion (ReviMo) project, founded by Aleksandr, aims to design and manufacture an innovative mobility device for people with disabilities. This initiative addresses the critical need for enhanced mobility solutions that promote independence and reduce caregiver dependency.

The project focuses on developing a user-friendly, remotely operable device adaptable to various environments. It includes stages from initial design and testing to prototype development and final product refinement. The expected outcome is a fully functional prototype leading to a commercially viable product, transforming the daily experiences of individuals with disabilities by offering new levels of autonomy and comfort.

Central to this project is the development of an intuitive and responsive motor control system. Utilizing the SVD48V motor driver and Raspberry Pi Pico, the system is operated via a joystick, providing a foundation for motion control in robotic mobility assistive devices.

1. Executive summary:

The Revival of Motion (ReviMo) project represents a significant step forward in enhancing mobility for individuals with disabilities through advanced technological solutions. Our team's focus was on developing motor control system utilizing a SVD48V motor driver and a Raspberry Pi Pico, designed to provide intuitive and responsive control for a robotic mobility assistive device. This system aims to significantly improve user experience and operational efficiency, embodying the project's commitment to promoting independence for mobility-impaired users.

A central achievement of our initiative was the integration of the Raspberry Pi Pico microcontroller with the SVD48V motor driver, which together form the backbone of the device's motion control capabilities. The microcontroller was effectively utilized as the central control unit, with a joystick connected to provide real-time, user-friendly directional input. This setup allows users to maneuver the mobility device with ease, enhancing their ability to navigate various environments autonomously.

The communication framework of our system employs the CAN bus due to its robust error detection and handling capabilities, which are crucial for reliable data transmission in dynamic and potentially challenging environments. Although the CAN communication link between the microcontroller and the motor driver is still in progress, we have established a functional preliminary control system using RS485 communication. This interim solution, managed through a PC-based software tool, has enabled us to achieve precise control and configuration of the motor driver, demonstrating the system’s potential even before the full integration of the CAN protocol.

However, the development process was not without challenges. The primary issue we faced was establishing a reliable CAN communication link, a complex task that required extensive troubleshooting and iterative testing. Our response to this challenge was to implement rigorous data integrity checks and develop efficient data handling techniques, ensuring stable and effective communication throughout the system.

The accomplishments of this project are significant, not only in demonstrating the feasibility of the proposed technology but also in laying a strong foundation for future enhancements. As we move forward, completing the integration of the CAN bus will be a priority, aiming to reduce system complexity and improve reliability. This work underscores our team’s commitment to innovation and our capacity to address and overcome technical challenges in pursuit of making a tangible difference in the lives of people with disabilities. Through this project, we contribute to the broader goals of improving accessibility and independence, aligning with societal efforts to support and empower individuals with disabilities.

2. Introduction and background:

The quest for enhanced mobility devices for individuals with disabilities has become increasingly urgent as we recognize the pivotal role such devices play in improving quality of life. While technological advancements have brought numerous innovations to mobility aids, significant gaps remain that affect the independence and dignity of users. This project, as part of the Revival of Motion (ReviMo) initiative, seeks to address these gaps by developing an advanced, user-centric mobility device designed to alleviate reliance on caregivers and enhance the autonomy of disabled individuals.

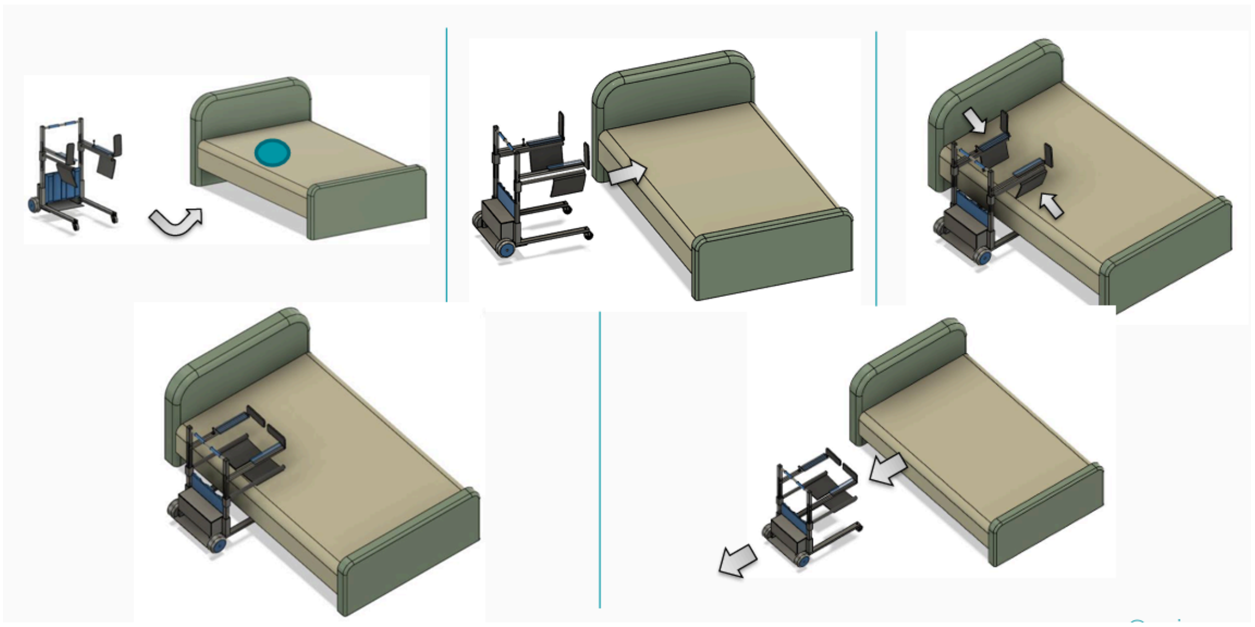
One of the primary issues with many existing mobility devices is the requirement for caregiver assistance. Users often depend on others to operate or navigate these devices, a dependency that can restrict their independence, impact their self-esteem, and limit their freedom. This reliance is not just a matter of convenience but a significant barrier to living a fulfilled and autonomous life.

Moreover, the physical operation of current mobility devices can be a challenge, particularly for those with limited strength or dexterity. The design of the controls, combined with the weight and the required effort to maneuver these devices, often makes them impractical for those who need them most. This not only discourages use but can exacerbate feelings of helplessness and frustration among users.

Privacy is another critical concern. The dependence on caregivers for mobility and daily activities can severely compromise the privacy of disabled individuals. This lack of privacy is not limited to public settings but extends into personal spaces, affecting basic movements and activities of daily living. Such intrusions can diminish a person's sense of personal space and autonomy, further impacting their mental and emotional well-being.

Additionally, the design of mobility devices often overlooks the toileting needs of users, making an already sensitive process complicated and uncomfortable. This oversight can affect the dignity and hygiene of the user, posing not only physical challenges but also emotional distress. Such designs necessitate either complex modifications to the existing devices or continued reliance on caregiver assistance for such basic needs.

Addressing these challenges, our project aims to develop a mobility device that is not only technically advanced but also deeply empathetic to the needs of its users. By integrating innovative control systems, reducing physical operation difficulties, and respecting the privacy and dignity of individuals, we strive to transform the landscape of mobility aids. The end goal is to create a solution that enhances the daily experiences and independence of individuals with disabilities, offering them a new level of autonomy and comfort.



Operation of the mobility device

Figure 1 [1]

3. Design Problem and System Requirements

Design Problem Statement

The primary challenge in our project was to develop a wheelchair that enables elderly users to have autonomous control over their mobility. The key innovation was to incorporate a system that not only allows control via a joystick but also includes a lifting mechanism. This mechanism aids in transferring the user from a bed to the wheelchair seamlessly, enhancing the independence of elderly individuals with limited mobility（Figure 1）.

System of Requirements and Constraints

The design was constrained by the need to ensure safety, affordability, and reliability. The main constraints we faced were:

1. Budget Limitations: Cost efficiency was crucial, necessitating the selection of affordable yet reliable components.
2. Safety Requirements: The system had to minimize operational risk, particularly in reducing the turning radius to prevent accidents during maneuvers.
3. Technical Constraints: We needed a stable and fast communication system for controlling the various mechanical components.

Evolution of System Requirements

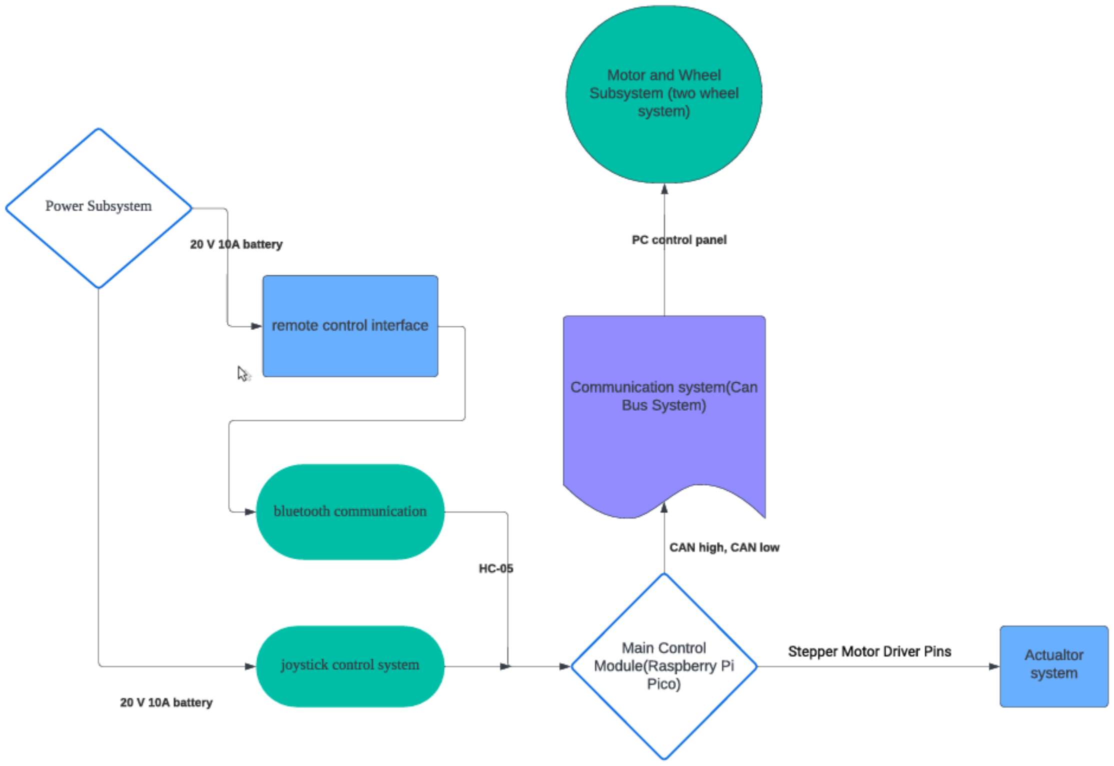
Initially, the project focused primarily on mobility through joystick control. However, as we gathered more feedback, the need for a safe and user-friendly lifting mechanism became apparent. Consequently, the requirements evolved to include robust support for lifting the user, leading to adjustments in the structural design and control system.

Review of Previous Work

Previous work in the field generally focused on mobility or lift-assistance separately. Our innovation was integrating these functionalities seamlessly, improving user autonomy. We also opted for a Raspberry Pi Pico for its cost-effectiveness and stable communication capabilities, a departure from more traditional, costlier solutions used in existing models.The development of what we did on communication could be seen at Figure 2.

Summary of Main Design Specifications

- Control System: Raspberry Pi Pico based for reliable and fast communication.
- Joystick Operation: Intuitive control system for easy maneuverability.
- Lifting Mechanism: Integrated lift to assist users in transferring from bed to wheelchair.
- Safety Features: Reduced turning radius to minimize mobility risks.
- Cost-Effective Design: Selection of components that balance quality and affordability.



Block diagram of operation subsystem

Figure 2

4. Solutions

Alternative Solutions

1. Remote Control: Implementing a system that allows users to independently operate the mobility device, such as a joystick-controlled system. This would reduce dependency on caregivers, enhance privacy and autonomy, and address the complicated toileting process.
2. Automated Seating Process: Integrating an automated seating mechanism to assist users in sitting and lying down. This is particularly beneficial for users with limited strength or dexterity, reducing the physical strain involved in using the device.
3. Regular Toilet Compatibility: Designing the mobility device to be compatible with standard toilets, ensuring maneuverability into bathroom spaces and appropriate positioning for toileting.
4. Different Communication Protocols: Besides CAN (Controller Area Network), other communication protocols like RS485 or RS232 could have been used. These are standard for serial communication and might offer easier integration with existing components.
5. Different Motor Types: Instead of changing the motors, adjusting the existing ones or selecting alternative types (like stepper motors or brushless DC motors) could have been an option, depending on the specific requirements of the device.
6. Different Control Methods: Aside from CAN communication, methods like PWM (Pulse Width Modulation) or analog inputs could have been used for controlling motor speed and torque.

7. Battery and Power Management Solutions: Given the operating voltage and current requirements, different battery management solutions could have been explored to optimize power efficiency and device autonomy.
Selected Approach and Rationale
1. The chosen approach involved changing to different motor models (SVD48V30A and SVD48V50A) and utilizing CAN communication for control. This decision was influenced by several factors:
2. Size Compatibility: The original motors did not fit the device. The selected models likely offered a better fit, enhancing the overall design and functionality.
3. Performance Requirements: The chosen motors support a range of powers (100-800W) and offer high torque (up to 50A), suitable for the device's needs.
4. CAN Communication: CAN is a robust, reliable communication protocol, particularly suited for automotive and mobility applications. It ensures efficient communication between different parts of the device, with less susceptibility to interference compared to other methods like RS232 or RS485.
5. Cost-Effectiveness: The prices of SVD48V30A and SVD48V50A ($160 and $180 respectively) were likely considered reasonable for the benefits they provide. The decision not to include an RS232-USB cable suggests a focus on cost optimization.
6. Software and Control Flexibility: These motors are compatible with SV-Config programming software and offer multiple control modes (speed, position, torque), providing flexibility in device operation. Remote control, particularly a joystick system, not only reduces dependency on caregivers but also enhances the user's independence and privacy. This is especially important for users with limited physical abilities.The automated seating process eases the physical burden on users while using the device, particularly beneficial for those with limited strength or dexterity. This combined approach is optimal for enhancing user independence and reducing physical strain, as it balances operational convenience with the physical needs of the users."

5. Design Implementation

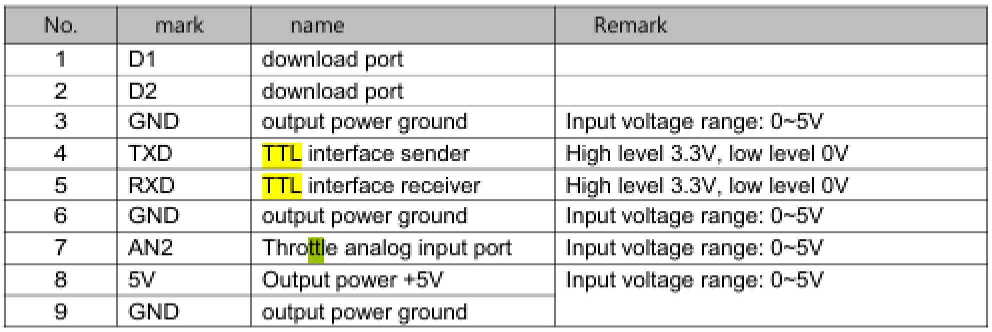
5.1 TTL Control Attempts

Introduction to TTL Control

In our pursuit to establish a foundational control system for the electric wheelchair, we initially turned to TTL (Transistor-Transistor Logic) signaling. TTL, known for its simplicity and effectiveness in digital circuits, was deemed a suitable starting point due to its widespread use and compatibility with a range of devices.

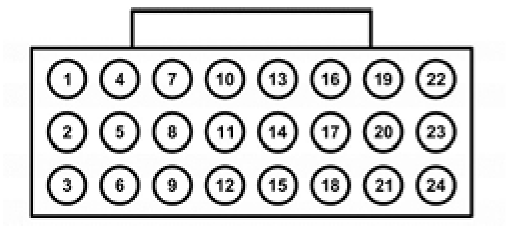
Implementing TTL Communication

We attempted to use 3.3V TTL for communication with the motor driver's interface. The motor driver’s interface included readily accessible TTL TX (transmit) and RX (receive) ports on its expansion interface, which was ideal for our initial tests. This setup was chosen because it matched our voltage requirements and was presumed to be straightforward for sending basic control commands to the motor driver.(Figure 3, 4)



No.	mark	name	Remark
1	D1	download port	
2	D2	download port	
3	GND	output power ground	Input voltage range: 0~5V
4	TXD	TTL interface sender	High level 3.3V, low level 0V
5	RXD	TTL interface receiver	High level 3.3V, low level 0V
6	GND	output power ground	Input voltage range: 0~5V
7	AN2	Throttle analog input port	Input voltage range: 0~5V
8	5V	Output power +5V	Input voltage range: 0~5V
9	GND	output power ground	

Figure 3 [2]



Extension ports

Figure 4 [2]

Challenges Faced

Despite the theoretical ease of using TTL for our application, practical challenges quickly surfaced. The main issue we encountered was the absence of predefined TTL communication messages that could directly interface with the motor driver to control its operations. Without the necessary command structures, our attempts to send control signals were unsuccessful.

Exploration for Solutions

We dedicated considerable effort to sourcing or creating compatible TTL command messages. This included reviewing the motor driver's documentation, seeking assistance from the device's technical support, and consulting with external experts who might have had similar integration challenges.

Potential for Future Success

The project did not yield immediate success with TTL communication due to the lack of appropriate control messages. However, this experience highlighted a potential path forward. If the necessary command protocols can be identified or developed, we believe that TTL could serve as a viable communication method for the electric wheelchair. This belief is supported by the compatibility of the physical connections and the proven reliability of TTL in similar applications.

Conclusion of TTL Attempts

While we did not achieve our goal of controlling the motor driver via TTL in this phase of the project, the foundational work we have done is promising. Future developments that include the discovery or creation of suitable TTL command sets could easily revive this approach as a feasible option for simple and effective motor control.

This expanded detail on the TTL control attempts provides a comprehensive view of our initial approach, the challenges we faced, and the potential for future success, encapsulating the iterative nature of engineering problem-solving.

5.2 CAN Bus Communication Tests Using Two Raspberry Pis

Introduction to CAN Bus Testing

To enhance the communication capabilities of our electric wheelchair project, we engaged with the Controller Area Network (CAN) bus, leveraging resources provided by Professor Hunter. The CAN bus system is renowned for its robustness in automotive and industrial applications, making it an attractive option for ensuring reliable communication between the various components of our wheelchair design.

Overview of CAN Bus

The CAN bus protocol facilitates multi-master communication, where multiple microcontrollers can communicate with each other without the need for a host computer. It is particularly valued for its high resistance to electrical interference and the ability to function over long distances with high integrity. This protocol uses a message-based system, allowing devices to transmit and receive messages without complex addressing schemes. Each message contains an identifier that indicates its priority, which is crucial in systems requiring real-time capabilities.

Setup for CAN Bus Testing

For our testing, we used two RP2040 microcontrollers, each equipped with a VP230 CAN transceiver module. The VP230 is a CAN transceiver that interfaces between the CAN protocol controller of the RP2040 and the physical wires of the CAN network. It is designed to withstand the harsh conditions of automotive environments, thus providing the reliability we needed for our tests.

Wiring Method

The connection setup involved linking the two RP2040 units via the CAN bus using the following wiring schema:

- CAN_H (High) and CAN_L (Low) lines of one RP2040 were connected to the CAN_H and CAN_L lines of the second RP2040.
- A termination resistor of 120 ohms was placed at each end of the bus to reduce signal reflections and ensure signal integrity.
- Both units were powered separately, and ground connections were shared between the two to provide a common reference point.

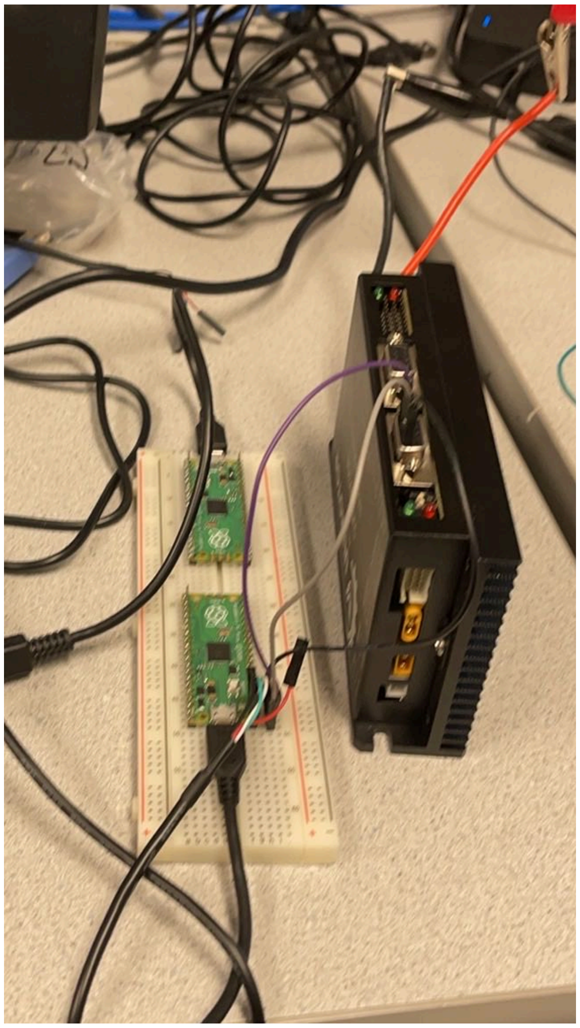
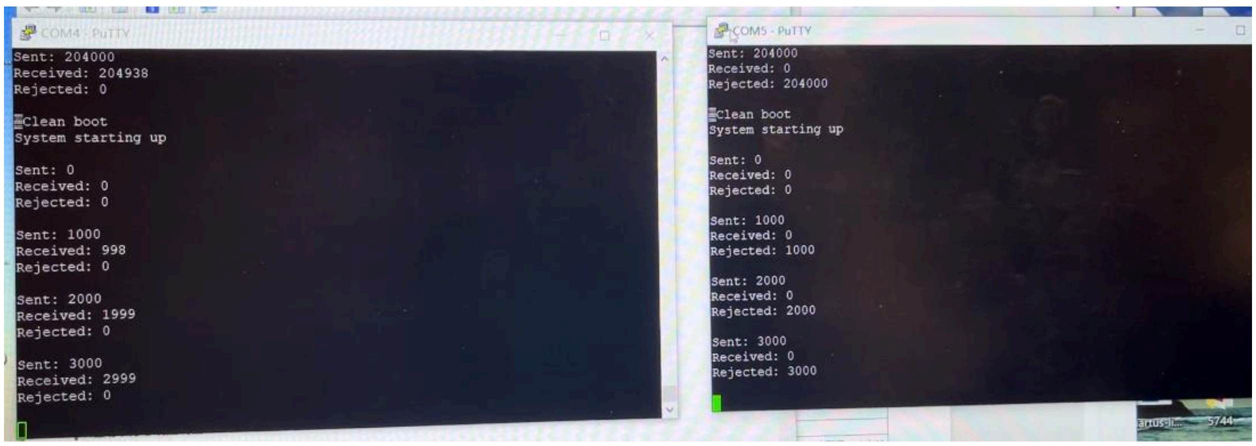


Figure 5

Testing Procedure and Results

Using the library and demos from Professor Hunter, we initiated a series of message sending and receiving tests. The tests were designed to evaluate the ability of each RP2040 unit to transmit and receive CAN messages effectively:

- Test Outcome: One of the RP2040 units successfully sent and received CAN messages, demonstrating the system's potential for reliable communication.
- Issue Encountered: The second RP2040 unit failed to participate in the communication. Upon investigation, it was determined that the VP230 CAN transceiver attached to this unit was defective. This malfunction prevented the unit from either sending or receiving any messages.



```
Sent: 204000
Received: 204938
Rejected: 0

Clean boot
System starting up

Sent: 0
Received: 0
Rejected: 0

Sent: 1000
Received: 998
Rejected: 0

Sent: 2000
Received: 1999
Rejected: 0

Sent: 3000
Received: 2999
Rejected: 0
```

```
Sent: 204000
Received: 0
Rejected: 204000

Clean boot
System starting up

Sent: 0
Received: 0
Rejected: 0

Sent: 1000
Received: 0
Rejected: 1000

Sent: 2000
Received: 0
Rejected: 2000

Sent: 3000
Received: 0
Rejected: 3000
```

Figure 6

Analysis and Conclusion

The successful communication by one of the units validates the choice of CAN bus for our application, highlighting its suitability for environments requiring robust and reliable data exchange. The failure of the second unit, while initially a setback, provided valuable insights into the importance of component integrity in the overall system reliability. Replacement of the defective VP230 module is expected to resolve the issue.

This detailed account of our CAN bus testing illustrates both the potential and challenges of implementing advanced communication systems in real-world applications, underscoring the need for thorough testing and quality control in component selection.

5.3 CAN Bus Control Trials

Integration of Motor Driver into the CAN Bus System

Building on the foundation laid by earlier communication tests, our project advanced to integrating the motor driver directly into our CAN bus system. This integration was critical, as it aimed to utilize the robust and reliable communication capabilities of CAN bus to control the wheelchair's motor effectively.

Setup for Motor Driver Integration

The connection involved interfacing the motor driver with our system via the CAN_H (High) and CAN_L (Low) channels, following standard CAN bus practices. This setup was intended to enable real-time control commands to be transmitted directly to the motor driver, which is pivotal for achieving precise maneuverability of the electric wheelchair.

Configuration and Testing Procedure

Following the motor driver's user manual, we programmed specific CAN messages to be sent to the motor driver. These messages were designed to control various aspects of the motor's operation, including speed, direction, and torque. The implementation process involved meticulous programming of the Raspberry Pi to craft and dispatch these commands via the CAN bus.

Challenges Encountered and Diagnostic Steps

Despite thorough preparations and prolonged testing, the communication with the motor driver was not successful. Initially, we suspected issues with the CAN messages themselves, possibly due to formatting errors or incorrect command parameters. To isolate the problem, we engaged several diagnostic tools and approaches:

- Message Verification: We used an oscilloscope to verify the presence of CAN messages on the bus. This test confirmed that messages were indeed being transmitted, indicating that the Raspberry Pi's message-sending functionality was operational.

- Signal Integrity Check: The oscilloscope tests also helped us confirm that the signal integrity on the CAN bus was maintained, ruling out hardware failures in the transmission path as a cause of the issues.

Potential Causes and Next Steps

Having established that our transmission hardware and setup were functioning correctly, we identified two likely sources of the communication failure:

- Incorrect Message Formatting: The most plausible explanation we considered was that the CAN messages might not have been formatted strictly according to the motor driver's specifications. Minor discrepancies in message structure or content could prevent the motor driver from recognizing or correctly interpreting the commands.
- Communication Protocol Mismatch: There was also a possibility that the method of crafting and sending messages via the Raspberry Pi might not fully comply with the motor driver's communication protocols, despite the messages being visible on the bus.

To address these issues, further steps were outlined:

- Engagement with Manufacturer: It became imperative to consult with the motor driver manufacturer's engineers to gain a deeper understanding of the specific communication requirements and possibly obtain example messages or more detailed programming guidance.
- Review of Documentation: A comprehensive review of both the motor driver's and the Raspberry Pi's CAN interface documentation was scheduled to ensure all settings and protocols were correctly implemented.

Following the initial diagnostic steps, we further adhered to the communication testing methods described in the motor driver's manual, sending a variety of message formats in an effort to establish a successful communication link. Despite these comprehensive efforts, the communication issues persisted, preventing effective control of the motor driver via the CAN bus.

Additional Testing Strategies

In response to these ongoing challenges, we have planned a new approach to diagnosing and resolving the communication problems:

- Hardware for Debugging: We intend to purchase specialized hardware that allows for CAN communication debugging using a computer. This hardware is designed to facilitate more detailed analysis and manipulation of CAN messages, which could provide greater insights into the issues at hand.
- Establishing Communication with a Computer: Initially, we will use the debugging hardware to set up a direct communication link between a computer and the motor driver. This step will help us verify the correct message formats and ensure that the motor driver is responding as expected without the additional variable of the Raspberry Pi.
- Re-testing with Raspberry Pi: Once we have confirmed successful communication between the computer and the motor driver, we will reintegrate the Raspberry Pi into the setup. This will allow us to isolate whether the issue lies within the Raspberry Pi's CAN message handling or elsewhere in the communication chain.

This phase of the project highlighted the complexities involved in setting up effective communication between different system components using CAN bus. The experience underscored the importance of precise alignment with hardware specifications and deep collaboration with component manufacturers. Further communication with the motor driver's manufacturer is expected to resolve these issues and enable successful integration of CAN bus control in our wheelchair project.

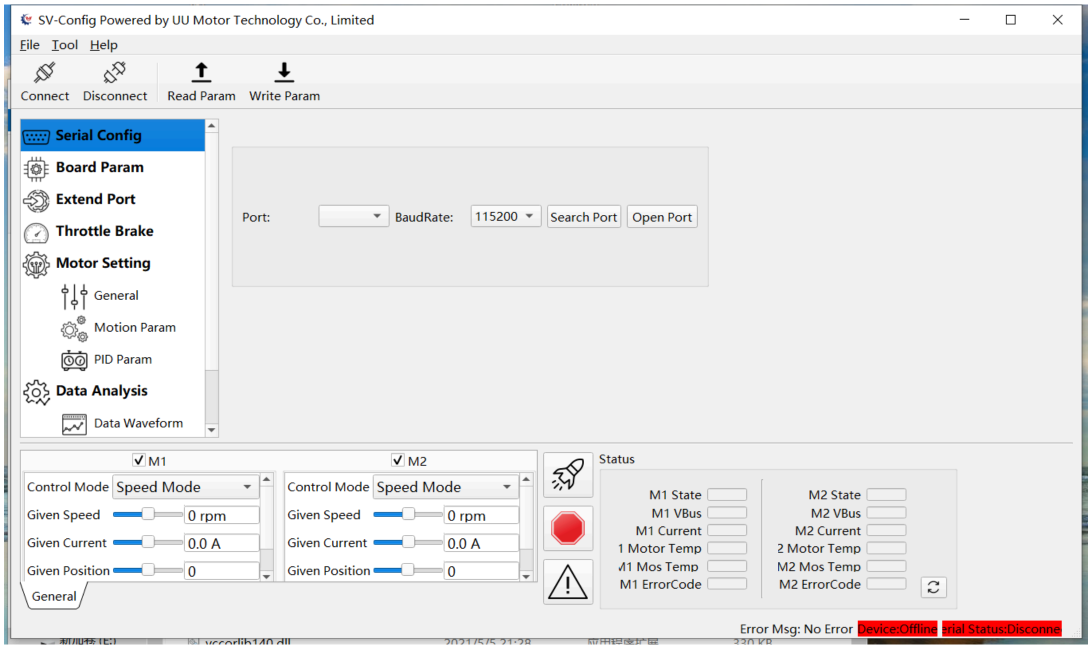
5.4 RS232 Connection for Parameter Adjustment

Introduction to RS232 Communication Setup

In an effort to enhance the configurability and tuning of the motor driver for our electric wheelchair project, we established a communication link using the RS232 protocol between the motor driver and a computer. RS232 was chosen for its straightforward implementation and reliability in serial communication, particularly suitable for setting and modifying device parameters directly from a computer.

Implementation of RS232 Communication

We utilized the motor driver's built-in software, which provided a comprehensive interface for adjusting various operational parameters. This setup not only facilitated easy modifications but also enabled detailed customization of the motor driver's settings.



Port:
BaudRate: 115200

M1		M2	
Control Mode	Speed Mode	Control Mode	Speed Mode
Given Speed	0 rpm	Given Speed	0 rpm
Given Current	0.0 A	Given Current	0.0 A
Given Position	0	Given Position	0

M1 State, M1 VBus, M1 Current, M1 Motor Temp, M1 Mos Temp, M1 ErrorCode
M2 State, M2 VBus, M2 Current, M2 Motor Temp, M2 Mos Temp, M2 ErrorCode

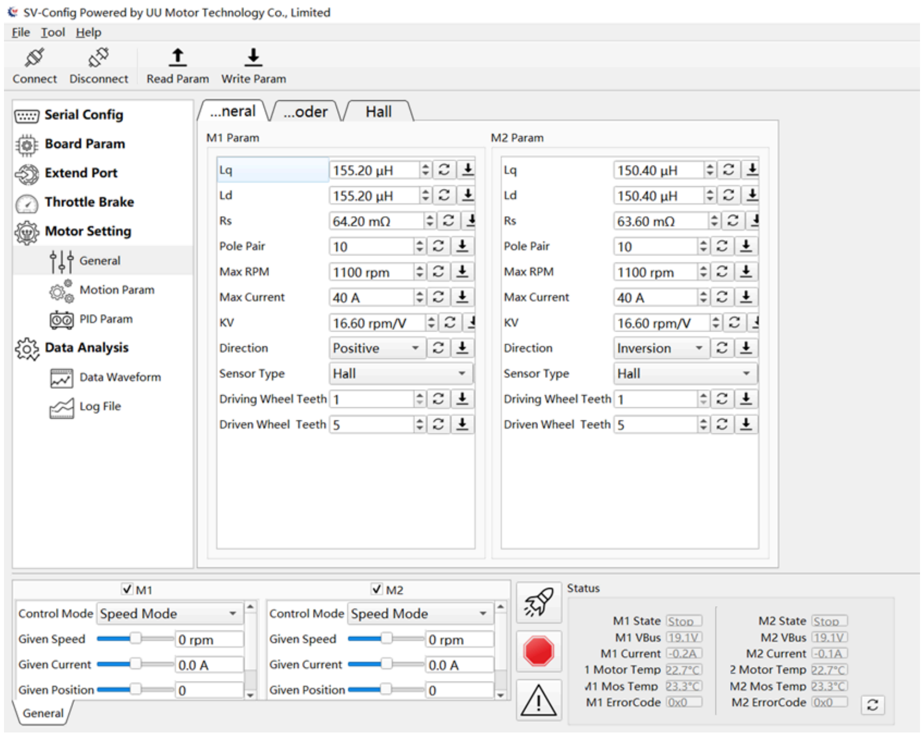
Error Msg: No Error Device:Offline Serial Status:Disconne

Figure 7

Configurable Parameters via RS232

The RS232 connection allowed us to access a wide range of motor driver settings, critical for optimizing performance according to the specific needs of our electric wheelchair:

- Torque Settings: Adjustments could be made to the motor’s torque output, ensuring that the wheelchair could handle different weight loads and terrain types without overloading the motor.
- PID Parameters: Fine-tuning the PID (Proportional, Integral, Derivative) controls was crucial for achieving stable and responsive control, essential for the safety and comfort of the wheelchair’s operation.
- Control Modes: We could switch between various control modes, such as speed control, torque control, or positional control, depending on the desired operational behavior.
- Communication Protocols: The RS232 interface also allowed us to configure the preferred communication protocols, aligning them with other components of the wheelchair’s control system.



M1 Param

Lq	155.20 µH
Ld	155.20 µH
Rs	64.20 mΩ
Pole Pair	10
Max RPM	1100 rpm
Max Current	40 A
KV	16.60 rpm/V
Direction	Positive
Sensor Type	Hall
Driving Wheel Teeth	1
Driven Wheel Teeth	5

M2 Param

Lq	150.40 µH
Ld	150.40 µH
Rs	63.60 mΩ
Pole Pair	10
Max RPM	1100 rpm
Max Current	40 A
KV	16.60 rpm/V
Direction	Inversion
Sensor Type	Hall
Driving Wheel Teeth	1
Driven Wheel Teeth	5

	✓ M1	✓ M2
Control Mode	Speed Mode	Speed Mode
Given Speed	0 rpm	0 rpm
Given Current	0.0 A	0.0 A
Given Position	0	0

Status

M1 State	Stop	M2 State	Stop
M1 VBus	19.1V	M2 VBus	19.1V
M1 Current	-0.2A	M2 Current	-0.1A
1 Motor Temp	22.7°C	2 Motor Temp	22.7°C
ʌ1 Mos Temp	23.3°C	M2 Mos Temp	23.3°C
M1 ErrorCode	0x0	M2 ErrorCode	0x0

Figure 8

Calibration and Hardware Testing via Computer

One of the key functionalities enabled by this setup was the ability to perform Hall sensor calibration directly through the computer. Hall sensors play a vital role in detecting the position of the motor rotor, which is crucial for precise motor control:

- Hall Sensor Calibration: Using the motor driver's software, we executed calibration routines that fine-tuned the motor’s response based on Hall sensor readings. This ensured that the motor's output closely matched the control inputs in terms of speed and position.
- Motor Operation Control: We also implemented control commands from the computer to the motor, which allowed us to test the motor’s response under various operational conditions. This was particularly useful for verifying the motor’s performance after adjustments and calibrations were made.

The RS232 connection proved to be a robust tool for parameter adjustment and real-time control testing. The ability to fine-tune and calibrate the motor driver from a computer significantly streamlined the development process, reducing the time and complexity involved in manual adjustments. Going forward, this setup will continue to serve as a critical component of our testing and configuration toolkit, ensuring that our motor driver is always operating optimally for the demands of the electric wheelchair.

5.5 RS485 Control Attempts

Introduction to RS485 Communication Challenges

In our quest to implement a robust communication system for our electric wheelchair project, we turned to RS485 due to its proven track record in industrial applications requiring long-distance and noise-immune data transmission. RS485's differential signaling offers superior resistance to electromagnetic interference, making it a potentially ideal choice for the electrically noisy environment of wheelchair control systems.

Initial RS485 Implementation Attempt

Our initial approach involved setting up an RS485 communication link using the RP2040 microcontroller to facilitate command transmission to the motor driver. The expectation was that RS485's capabilities would enhance the reliability of our control system across various operational scenarios.

Challenges with RS485 Hardware

Despite the theoretical advantages, our implementation faced significant practical challenges:

- Lack of RS485 Expansion Board for Raspberry Pi: We discovered that the RS485 communication could not be effectively established because we lacked an appropriate RS485 expansion board for the Raspberry Pi. This board is crucial as it converts the TTL signals from the Raspberry Pi to RS485 differential signals, which are essential for proper communication over longer distances and through noisy environments.
- Insufficient Communication Voltage: The absence of a suitable expansion board resulted in inadequate voltage levels for RS485 communication. RS485 standards require specific voltage differentials to ensure reliable data transmission, and without the proper hardware, these levels were not achieved, leading to failed communication attempts.

Decision to Halt Further RS485 Trials

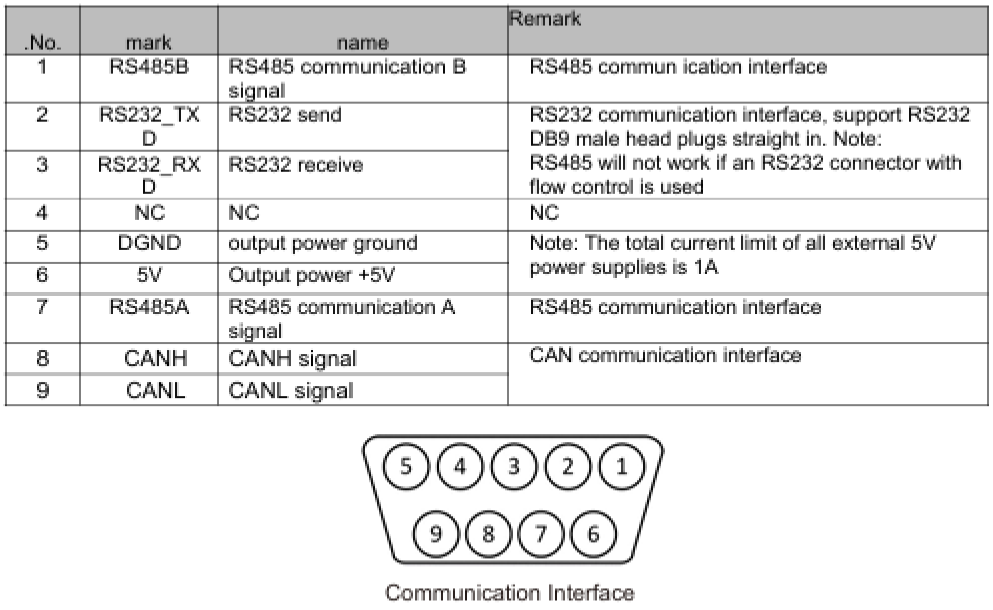
Given the hardware limitations and the initial failure to establish a stable communication link:

- Halting RS485 Trials: We decided to halt further attempts with RS485 until we could secure the proper expansion hardware. Continuing with inadequate setups would likely result in further failures and could potentially damage the existing equipment.

Diagram and Protocol Overview

At this point, a diagram illustrating the intended RS485 communication setup and protocol would be useful to visualize the planned configuration and the specific points of failure. However, as the diagram was mentioned but not included in your text, detailing it here would involve describing the typical RS485 communication topology:

- Two-Wire Configuration: Typically, RS485 communication uses a two-wire setup where each device on the network can both send and receive data, making it ideal for multi-device communication.
- Differential Signaling: This method uses two wires (A and B) for each signal, with data integrity maintained by comparing the voltage difference between these two wires, rather than referencing a ground.



.No.	mark	name	Remark
1	RS485B	RS485 communication B signal	RS485 commun ication interface
2	RS232_TXD	RS232 send	RS232 communication interface, support RS232 DB9 male head plugs straight in. Note: RS485 will not work if an RS232 connector with flow control is used
3	RS232_RXD	RS232 receive	
4	NC	NC	NC
5	DGND	output power ground	Note: The total current limit of all external 5V power supplies is 1A
6	5V	Output power +5V	
7	RS485A	RS485 communication A signal	RS485 communication interface
8	CANH	CANH signal	CAN communication interface
9	CANL	CANL signal	

Figure 9 [2]

Next Steps and Future Plans

To resolve the issues encountered and potentially revive the RS485 communication strategy, our future plans include:

- Acquisition of RS485 Expansion Boards: We will procure suitable RS485 expansion boards that are compatible with the Raspberry Pi. This will ensure that the communication voltage requirements are met and that the system adheres to RS485 standards.

- Comprehensive Testing Post-Setup: Once the appropriate hardware is in place, rigorous testing will be conducted to validate the communication setup under various operational conditions, ensuring that the system is robust and reliable.

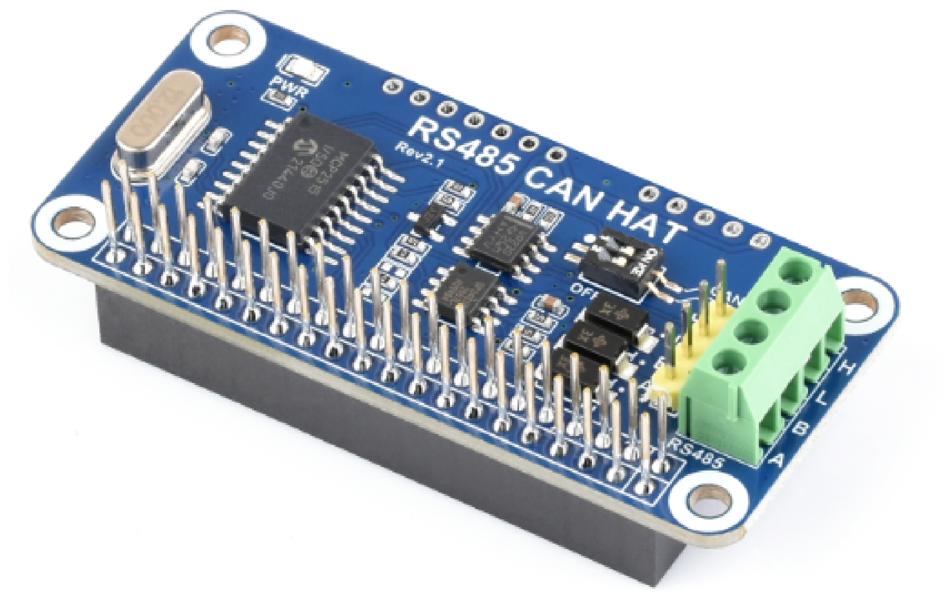


Figure 10 [3]

5.6 Introduction to Joystick Implementation

In our electric wheelchair project, implementing a joystick interface was crucial for providing a user-friendly and intuitive control system. The joystick allows for manual control of the wheelchair's direction and speed by translating the user's physical movements into digital signals that the control system can interpret.

Setup and Configuration

1. Joystick and RP2040 Connection:
   - The joystick was equipped with four distinct output lines corresponding to different directions (forward, backward, left, right).
   - Each output was connected to a specific GPIO (General Purpose Input/Output) pin on the RP2040 microcontroller.
   - A ground (GND) connection was also established between the joystick and the RP2040 to complete the circuit and provide a reference point for the signal voltages.
2. Configuration of GPIO Pins:

- The GPIO pins on the RP2040 were configured as digital inputs. This setup was necessary to read the high (activated) or low (deactivated) signals from the joystick corresponding to user movements.
- Each pin was programmed to detect changes in state, triggering an interrupt that would initiate the appropriate response from the motor controller.

Testing and Implementation

1. Functional Testing:
    - Initial tests were conducted to ensure that each GPIO pin was correctly reading the signals from the joystick. This involved manually manipulating the joystick in all directions and verifying that the corresponding pins registered the changes accurately.
    - The functionality of the joystick was tested in a controlled environment to ensure that all possible movements could be accurately captured and translated into commands.
2. System Integration:
    - After successful standalone testing, the joystick was integrated into the larger control system of the wheelchair.
    - Additional tests were carried out to ensure that commands from the joystick were correctly interpreted by the RP2040 and relayed accurately to the motor driver, resulting in the desired movements of the wheelchair.
3. Performance Evaluation:
    - The responsiveness and accuracy of the joystick controls were evaluated in various scenarios, including different surface types and inclines, to mimic real-world conditions.
    - Adjustments were made to the sensitivity and calibration of the joystick inputs to ensure optimal control over the wheelchair.

The successful implementation and testing of the joystick interface have significantly enhanced the control system of our electric wheelchair project. The RP2040 microcontroller efficiently processes the joystick inputs, providing precise and responsive control to the user. This setup not only meets our project's requirements for an effective manual control mechanism but also ensures that the wheelchair can be operated easily and safely under various conditions. This phase of the project underscores our commitment to user-centric design and accessibility.

6.Revised Testing Strategy for Universal Communication Protocols

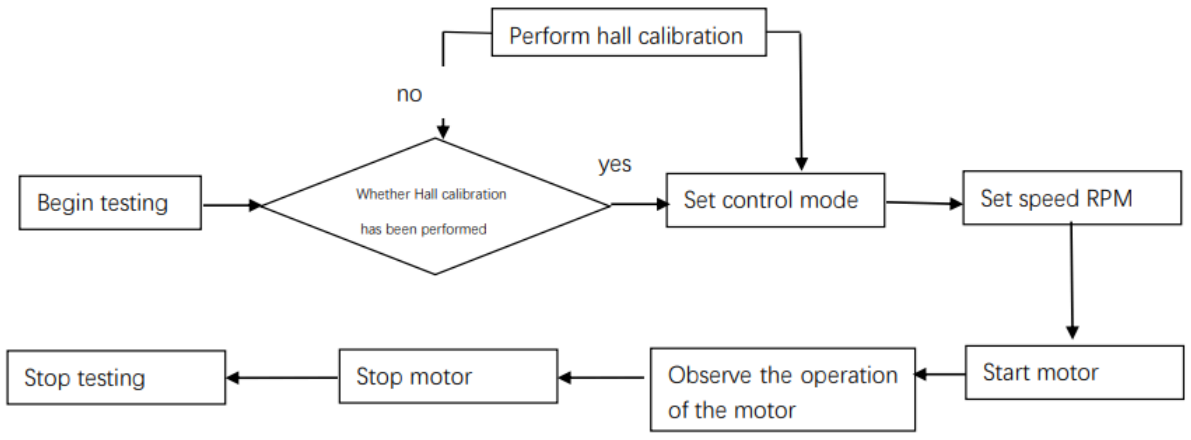
Overview

The provided flowchart outlines a general testing strategy applicable to various communication protocols used in the control system of an electric wheelchair. This universal strategy is designed to assess the efficacy of communication methods, including but not limited to RS485, RS232, CAN, and others potentially integrated into the system. The primary focus is on verifying communication reliability, command execution, and system response to different operational scenarios.

Test Setup

1. Equipment Preparation:
    - Set up the necessary components including the wheelchair's motor, motor driver, control unit (e.g., RP2040), and the chosen communication modules (RS485, RS232, CAN, etc.).
    - Ensure all connections are secure, and the software or firmware for each component is updated to the latest version.
2. Environment Setup:
    - Prepare a testing environment that simulates real-world conditions expected during the wheelchair's operation.
    - Include potential interference sources if testing robust protocols like RS485 to evaluate noise immunity.

Test Execution



1. Initial Communication Test:
    - Begin with basic communication checks to ensure that each component can send and receive messages.
    - Verify the basic operational functionality such as turning the motor on and off.
2. Command Execution and Verification:
    - Send commands to adjust motor RPM and other operational parameters.
    - Use instrumentation, such as a tachometer or software logs, to confirm that the motor output matches the command specifications.
3. Error Handling and Recovery:
    - Introduce errors intentionally to test the system's error detection and handling capabilities.
    - Evaluate the system's ability to revert to safe states or send appropriate alerts.
4. Feedback and Adjustment Loops:
    - Assess the system's feedback mechanisms by altering operational conditions dynamically and observing the response.
    - Adjust parameters through manual controls and automated responses to evaluate the system's adaptiveness.

Long-term Stability and Stress Tests

- Conduct extended tests to monitor the system's performance over time, focusing on aspects such as durability, consistent response under continuous use, and recovery from potential failures.

Documentation and Analysis

- Document all test cases, results, and observations meticulously.
- Analyze data to identify patterns or recurring issues that may indicate deeper systemic problems.

This comprehensive testing strategy is designed to validate communication reliability across different protocols integrated into the electric wheelchair's control system. By thoroughly testing each component and the system as a whole, we aim to ensure that the wheelchair is reliable, safe, and responsive under all expected conditions. This strategy will also aid in pinpointing areas that require further refinement or additional testing.

Each of these methods was explored as a separate entity, providing valuable insights into the possibilities and limitations of different communication and control technologies. Although none of the methods fully met all the criteria for optimal wheelchair control on their own, they each contributed significantly to our understanding of what would be required in a composite solution.

Acknowledgements

We would like to express special thanks to our advisor, Dr. Hunter Adams for his tremendous help and guidance throughout this project!

We also thank the ECE Department, Aleksandr Malashchenko, for their help on this project.

Reference

[1] Aleksandr Malashchenko, “ReviMo Rev Protofacturing”

https://www.revithaca.com/

[2] SVD48V30A-user-manual-V2.0

https://www.uumotor.com/

[3] RS-485 Shield for Raspberry Pi

https://www.electromaker.io/shop/product/rs-485-shield-for-raspberry-pi?gad_source=1&gclid=CjwKCAjwrvyxBhAbEiwAEg_KgoTl2z-0PwB9y6DsfcEapWsMoKLn5M-XhuzXVF96ECk1bjZD6al8AhoC53IQAvD_BwE