Boston University
Electrical & Computer Engineering
EC464 Capstone Senior Design Project

User's Manual

MOSS Wheelchair

Submitted to

Prof. Osama Al Shayk
9 St. Mary's St.
Boston, MA

by

Team 11

Team Members

Sebastian Gangemi sgangemi@bu.edu
Aya Kassem ayak@bu.edu
Norbert Malik norbert@bu.edu
Maha Noor mahakdn@bu.edu

Submitted: 15 April 2024

MOSS Chair User Manual

Table of Contents

(Right-click on Table of Contents to update fields and page numbers automatically)

Executive Summary - Maha

Around 1% of the world's population needs wheelchairs, but a basic wheelchair cannot meet everyone's needs and expectations. Therefore, we are proposing to build a Modular Open Source Smart (MOSS) Wheelchair in which engineers will be able to create modules for a user's wheelchair as needed. For our project, we will implement three modules while also creating an open-source website for easy access to our modules. The first module introduces a 3D-printed robotic arm implemented with servo motors for precision in movement and manipulation. The second module features navigation, where we employ computer vision. This algorithm facilitates advanced real-time object detection and recognition, allowing the wheelchair to navigate its environment intelligently. The third module tackles obstacle detection and avoidance, utilizing the outputs of computer vision to intricately control the motors and electronics responsible for wheel movement and robotic arm functionality. Using computer vision will allow our wheelchair system to gain an enhanced ability to interpret its surroundings, offering users a more sophisticated and adaptive mobility solution. Through this comprehensive approach, we aim to redefine wheelchair capabilities and foster inclusivity while providing users with customizable, state-of-the-art mobility solutions.

1 Introduction - Maha

According to the WHO, there are 65 million wheelchair users in the world, each with unique functional needs. Current products are often tailored to specific disabilities, making it challenging for companies to address the diverse demands within this niche market efficiently. Additionally, many smart wheelchairs are extremely expensive and difficult to understand and even operate. The purpose of the MOSS Chair is to create an open-source platform for smart wheelchairs. This platform will allow developers to implement modular designs, offering a continuously expanding range of features to cater to the varying needs of wheelchair users.

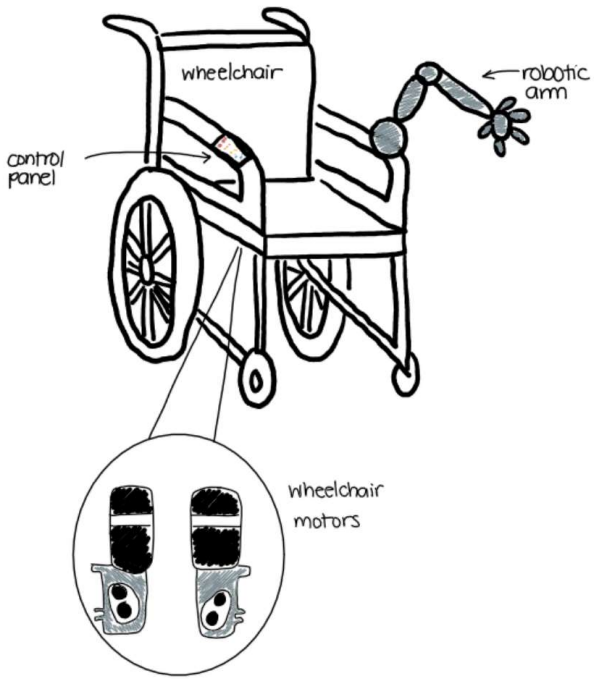
Our approach solves the user's problem by offering an open-source platform with modular designs. The wheelchair will initially feature assistive navigational functions, motorization for autonomous capabilities, computer vision algorithms like Inception-V2 for reliable and fast object detection, and compatibility with a programmable assistive robotic arm. This modular design allows developers to tailor features according to specific user needs, providing a customizable and adaptive solution. Additionally, our project involves community collaboration, which encourages a community of developers to contribute and innovate, expanding the range of available features over time.

Each component chosen to solve this problem has its purpose. The decision to prioritize modularity and open-source principles is crucial in effectively addressing the various aspects of the customer's problem. Modularity ensures flexibility, allowing the wheelchair to adapt to varying user needs, while open-source principles encourage collaboration and innovation within the community. The chosen conceptual approach stems from the recognition that a rigid, closed-system approach would fall short in addressing the dynamic and diverse landscape of mobility challenges. By adopting a modular, open-source framework, we lay the groundwork for a solution that not only meets present needs but also invites ongoing collaboration, adaptation, and innovation to cater to emerging user requirements.

Our design implements the NVIDIA Jetson Nano as the control hub which handles signal processing for multiple different processes in the design. It utilizes PWM signals to control the Arduino Nano that communicates with the wheel’s motor encoders in order to move the wheelchair in the desired direction and speed. Additionally, the Jetson Nano uses computer navigation, with images coming from the connected Raspberry Pi NoIR camera and LiDar sensor, to move the robotic arm to its desired destination.

The Jetson Inference library that we used contained a pre-trained Inception-V2 model trained on Microsoft’s COCO dataset, which features multiple instances of common objects. We imported and trained this model on additional datasets and used it to detect objects in real-time from the Raspberry Pi SC0024 IMX219 NoIR Camera connected to the Jetson Nano.

All in all, with the combination of sensors, robust motors and the powerful Jetson Nano, the MOSS Chair serves as a sustainable and accessible replacement for the traditional smart wheelchair.

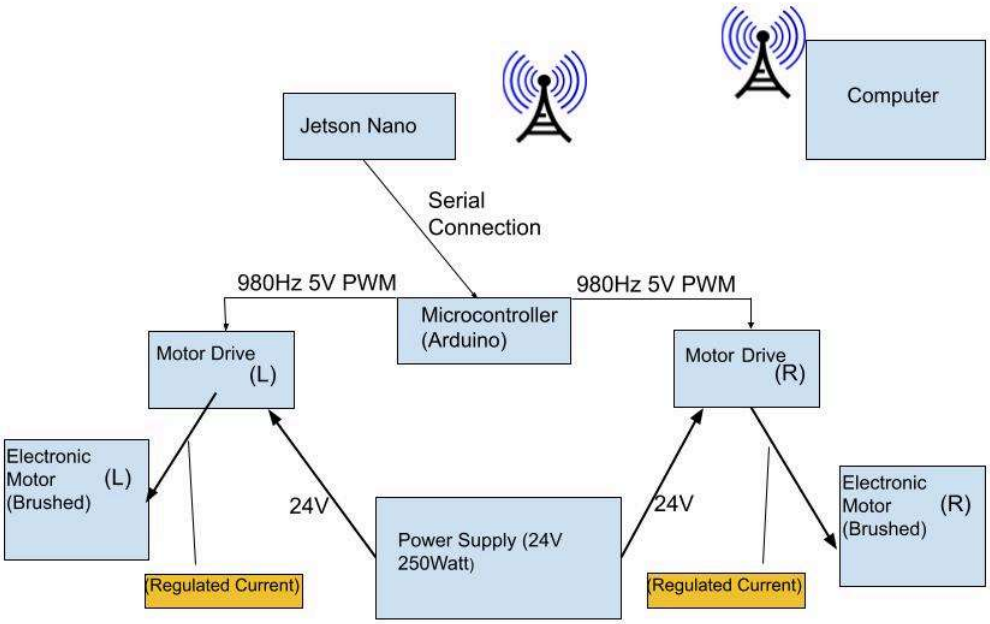


This drawing shows the overall anatomy of the smart wheelchair.

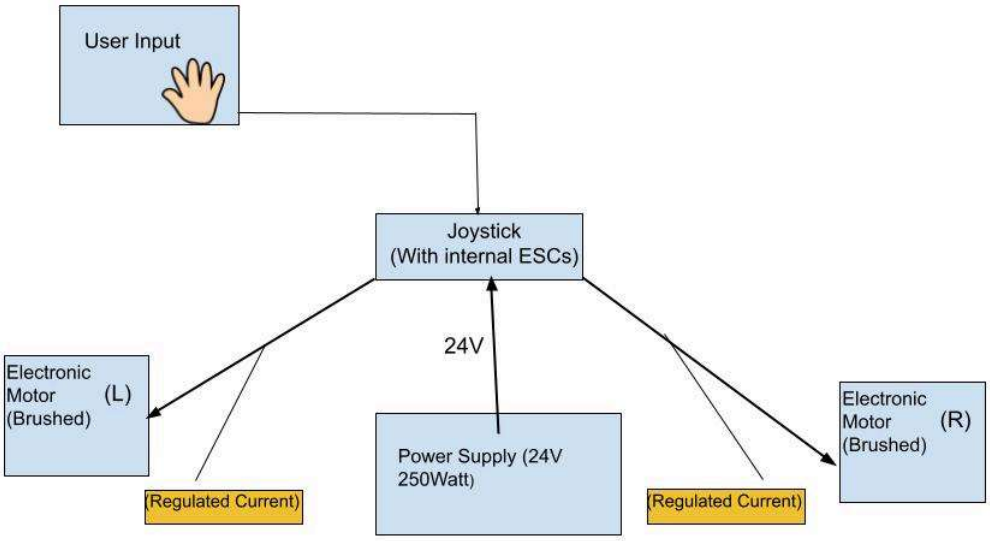
2 System Overview and Installation

2.1 Overview block diagram - Sebastian

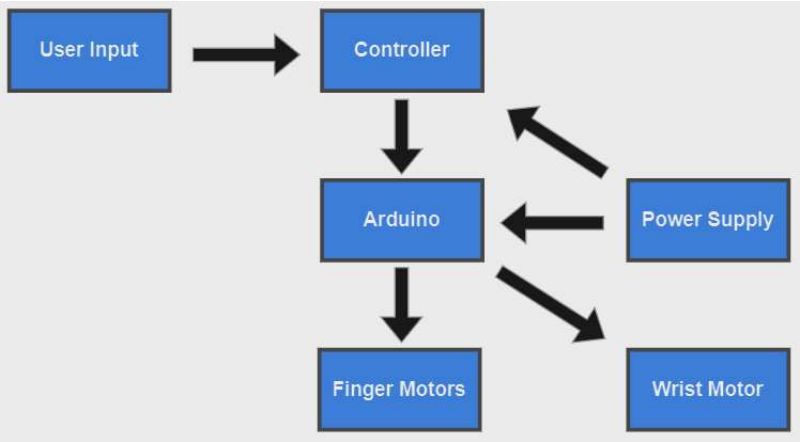
Remote Navigation:



Joystick Navigation - Sebastian:



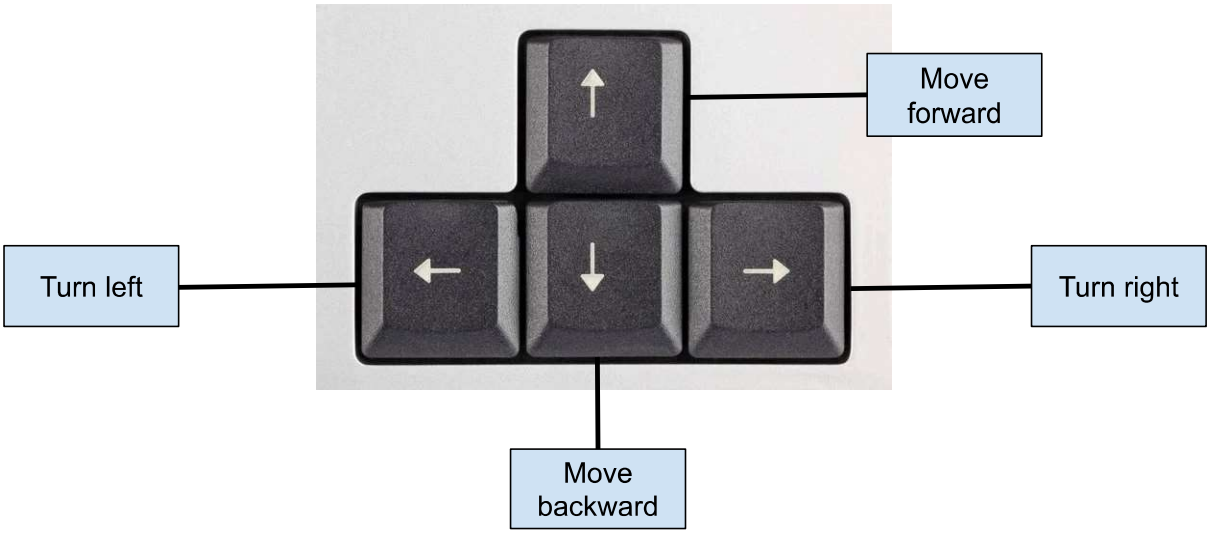
Arm Control - Nobert



2.2 User interface.

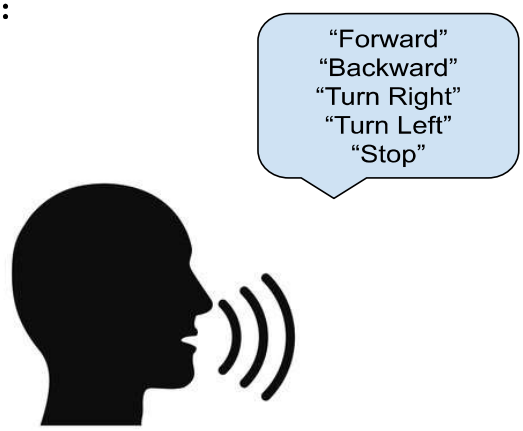
Controlling interface - Sebastian:

Laptop Remote Control Interface - Aya:



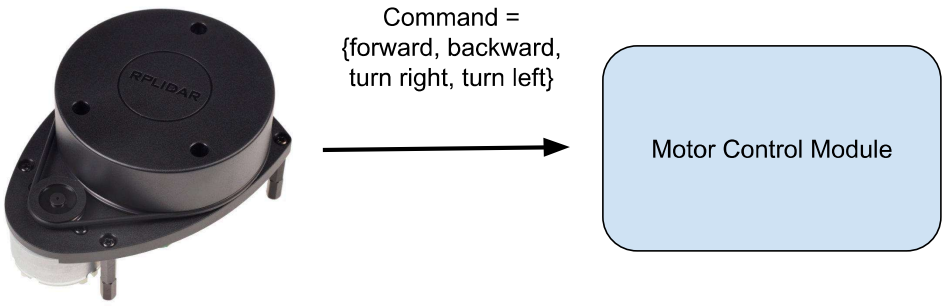
Note: Chair will keep moving forward/backward/right/left (depending on the command given) until the key is released.

Voice Control Interface- Aya:



Note: The chair will keep moving forward/backward/right/left or stay stopped (depending on the command given) until another command is given.

LIDAR Control - Aya:

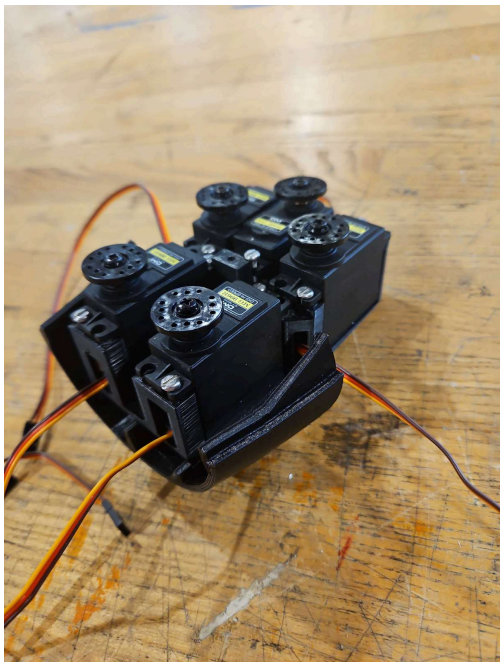


2.3 Physical description.

Overall Wheelchair with Navigation System - Maha:

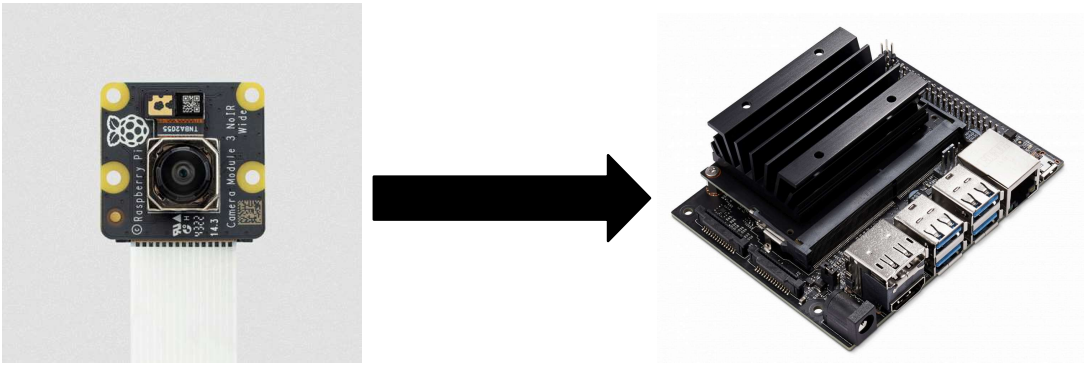
The image above shows the back of the wheelchair with the added navigation module. A piece of plywood can be seen (which is detachable). On the plywood, the motors, controllers, Arduino, Jetson Nano, are batteries are attached which contribute to the movement of the wheelchair.

Arm- Norbert:



The images above show the constructed arm, with the image on the right depicting the motors that are housed inside the arm, underneath what is marked in the image on the left as part 17.

Object Detection - Aya:

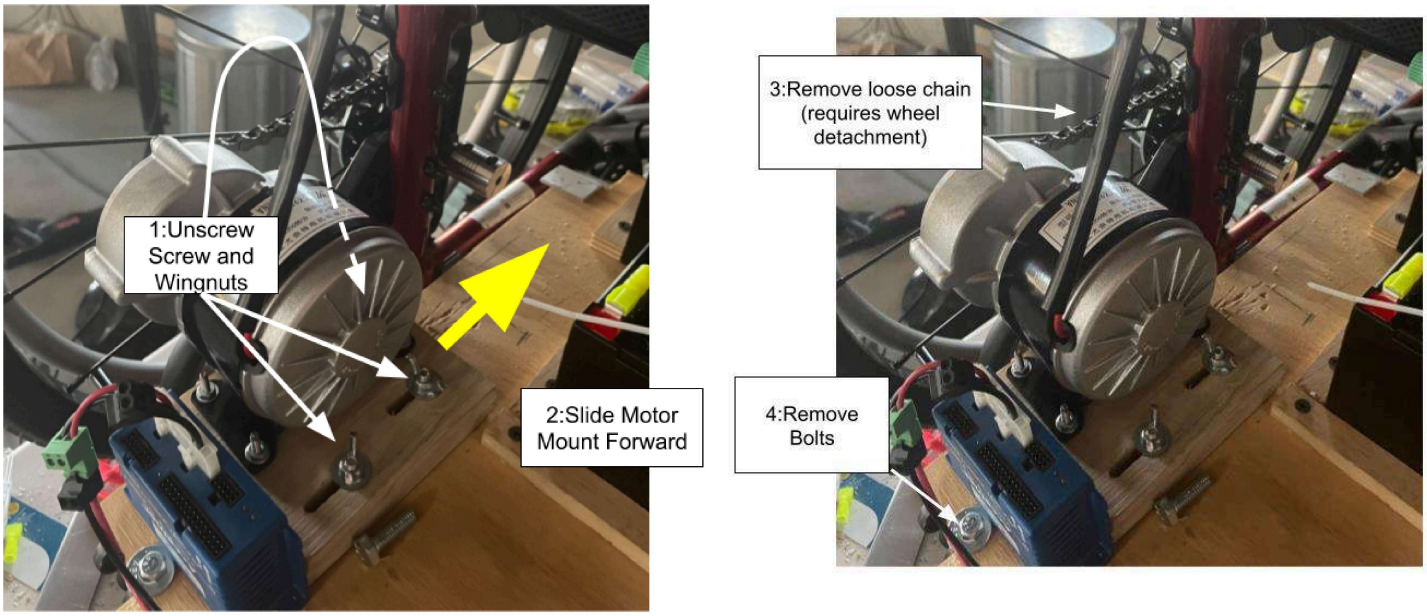


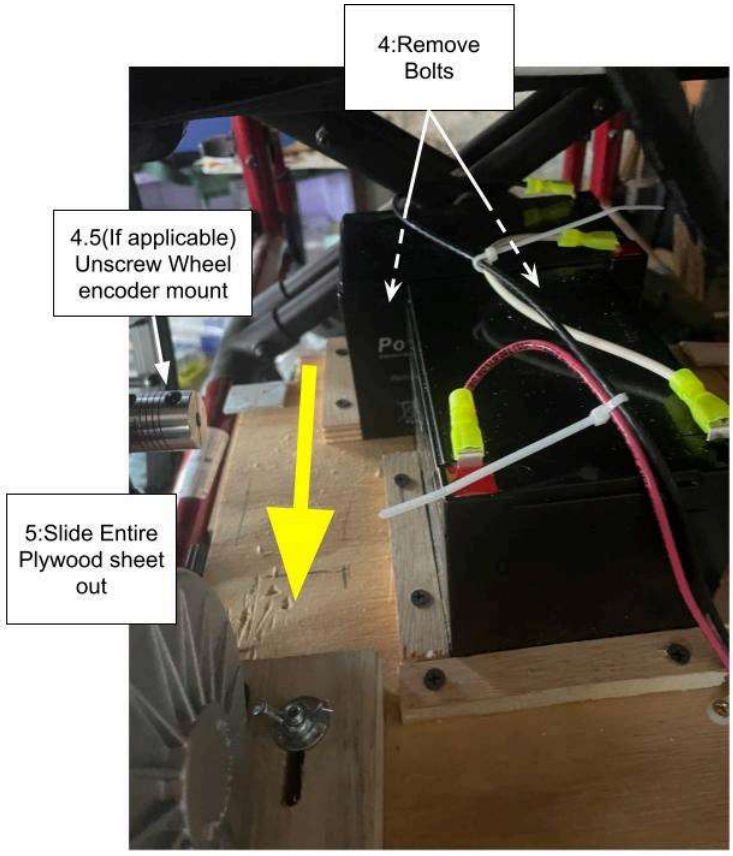
The object detection hardware involves a Raspberry Pi SC0024 IMX219 NoIR Camera connected to the Jetson Nano. The camera takes in real-time input and sends it to the Jetson Nano for processing.

2.4 Installation, setup, and support

Motor System - Sebastian:

The motor system's mounting is modular and detachable.





Reattach in reverse steps of detachment:

1. Slide Plywood In
2. Screw in Bolts
3. Attach Chain
4. Slide Motor mount backwards (until chain is taught)
5. Tighten Wingnuts

Arm:

1. Arm unit is fully assembled, so no changes to it are needed
2. Arm needs to be screwed onto the platform
3. Motor Bed and Arduino need to be screwed onto platform behind arm
4. Connect Arduino to Motors in Bed
5. Connect Controller to Arduino
6. Connect Power supply to Arduino

Control System Set Up:

Jetson Nano - Maha:

1. Acquire SD card and appropriate voltage regulator with micro USB cable (output requirements = 5 V, 2.5 A) for Jetson Nano
2. Follow the setup and installation procedure as specified in the following website: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro
3. Download appropriate files as needed.

Note: Power is always last to plug in and first to unplug

Joystick Control - Sebastian:

1. Connect Left and Right Motors to corresponding output cables.
2. Connect 24V power Supply To joystick (with internal ESC.)

Laptop Remote Control - Aya and Sebastian:

1. Power on Jetson Nano, SSH into it
2. Connect left motor to left motor drive, connect right motor to right motor drive
3. Connect 24V power supply
4. Run “control_chair_keys.py”
5. Control the MOSSChair with the arrow keys of your laptop

Speech Control - Aya:

1. Power on Jetson Nano, SSH into it
2. Connect left motor to left motor drive, connect right motor to right motor drive
3. Connect 24V power supply
4. Run “control_chair_speech.py”
5. Give the chair commands out loud. Commands include:
    a. Forward
    b. Backward
    c. Turn right
    d. Turn left
    e. Stop

LIDAR Control - Aya:

1. Power on the Jetson Nano, SSH into it
2. Connect left motor to left motor drive, connect right motor to write motor drive
3. Connect 24V power supply
4. CD into the ROS workspace: `cd catkin_ws`

Arm Control - Norbert:

1. Connect Motors to Arduino
2. Connect Arduino to Controller
3. Connect Arduino to Power Supply

Object Detection -Aya:

1. Power on the Jetson Nano, SSH into it
2. CD into object detection directory: `cd object_detection`
3. Run the inference script with pre-trained Inception-V2 model: *python3 inference.py*

3 Operation of the Project

3.1 Operating Mode 1: Joystick - Sebastian

The Joystick acts as one of our manual forms of operation, and is suitable for users with function in their hands and arms. This Joystick comes preinstalled with ESC and acts as the easiest/quickest way to test the structural integrity/mechanical success of the physical design. Max torque is adjustable, though depending on the rigidity of the motor's mounting system and the potential presence of a gusset, higher speeds(torques) may not be attainable without damaging the system.

3.2 Operating Mode 2: Laptop Remote Control - Aya

This operating mode is also another manual form of operation suitable for users with function in their hands and arms, but it provides the user more freedom and control in the sense that, unlike the joystick, the laptop does not have to be connected to the chair and the operation of the chair is completely wireless. This can be useful for cases where the user needs to move the chair around when they are not sitting in it (such as to bring it closer to them when they want to get on it or when they want to move it away from their space when they are not using it, etc.). Functions of this mode include going forward, backward, turning right, or turning left depending on what arrow key of a laptop is being held down. The chair will keep listening to and executing the given command until the key being held down is released. Another main difference between this mode and the Joystick Control mode is that the Laptop Remote Control currently only supports the chair going at constant preset speeds with the chair's acceleration set to 0.

3.3 Operating Mode 3: Speech Control - Aya

This operating mode is a reach goal for this project that we started implementing and we hope future teams can build upon and finish implementing. Unlike the previous two modes, this mode is suitable for users who do not have the capability of using their hands and arms. This mode is very similar to the laptop remote control as it uses the same motor control code with the same backend functions to send the chair forward, backward, right, or left, or make it do a full stop. It also currently only supports preset constant speeds (we can vary this preset speed, but the acceleration is set to 0). Instead of taking in input by listening to the arrow keys of a laptop, the input is taken by listening to the user's voice and waiting to recognize commands via a USB Microphone connected to the NVIDIA Jetson Nano. This places a big emphasis on the "Modularity" goal of this project by taking in a variety of forms of inputs that can be "plugged" into the main control module of the project.

3.4 Operating Mode 4: LIDAR Control - Aya

This operating mode is also one of our reach goals for this project. The idea behind this is to have an autonomous mode where the decision to move forward, backward, turn left, or turn right is completely determined by an algorithm built on the Robot Operating System

(ROS) that takes in the input of a 360 degrees RPLidar sensor and sends the appropriate commands to the motors using the same motor control module used by previous modes. Despite not having the LIDAR Navigation Control fully implemented, we have built ROS on the NVIDIA Jetson Nano and set up the LIDAR sensor to be able to detect objects around it. We have laid the foundations for this additional operating mode with the plan of future teams building on top of it in mind.

3.5 Robotics Arm + Object Detection - Norbert and Aya

The robotic arm is operated by a group of motors, each controlling one finger and the wrist. Each of the motors can be used individually or in tandem with any number of the other motors. The string tendons that run inside the arm are wound together at the attachment at the motors, resulting in a constant tension force in the fingers, allowing them to resist forced opening and closing. The motors are operated through an Arduino that can receive instructions from a computer. It could also be coded to receive instructions from a joystick or touchpad, but this was not done as there is no space to integrate such a system on the wheelchair as of now. The robotic arm on our wheelchair is accompanied by an object detection module designed to identify objects for interaction, such as picking up items or pressing buttons. This capability, similar to the LIDAR Control Mode, was one of our ambitious goals that given time constraints, we were not able to achieve. We envisioned it as a foundational technology that future teams could further develop and refine.
To facilitate this, we have equipped the wheelchair with a Raspberry Pi SC0024 IMX219 NoIR Camera, which is adept at capturing images even in low light conditions. This camera is connected to the NVIDIA Jetson Nano, forming a hardware setup capable of processing complex visual data in real-time.
In addition, we have integrated an Inception-V2 model into the Jetson Nano's computing framework. This model has been pre-trained on Microsoft's COCO dataset, a rich collection of images that include multiple instances of everyday objects. The choice of the Inception-V2 model leverages its high accuracy in object detection without compromising speed, making it ideal for dynamic environments where the wheelchair and its robotic arm need to interact seamlessly with a variety of objects.

3.6 Safety Issues - Sebastian, Maha, Norbert, and Aya

- Power is ALWAYS last to plug in and first to unplug. Other orders of disconnection may result in unintended movement in motors.
- Not suitable for torques large enough to allow inclined travel and higher speeds, would need gusset implementation to achieve such torques without buckling of the motor mount.
- Must not move wheels by hand while motors are plugged in and power is unplugged, this may generate dangerous EMF in the motor. Rule of thumb: unplug motors immediately after unplugging power so you don't forget.
- Note that when commanding a speed of 0 onto wheels, if wheels are moved by hand the wheel will attempt to return to its original position. This may cause erratic behavior at higher-than-expected speeds.

- When in speech control, someone in the user's surroundings may pronounce commands out loud. This would make the wheelchair execute the command without the user's consent.
- Due to the large voltage of the battery, it is imperative to keep the ground and hot wires away from each other at all times. Any contact could result in arc flashes and bodily damage.
- Pins may become loose over time due to excessive movement, which can cause erratic behavior. Check all connections before use to ensure full safety.
- The arm may attempt to perform a task it is not suitable to, such as holding a weight too great, which would most likely cause the motors to fail. This would not be fixable by the individual themselves and could make the arm dangerous to operate.
- While unlikely, a structural failure in the arm could send shards of plastic or metal bits flying, potentially hitting the users or others.
- The arm could get in the way of the wheelchair, or get into the wheels, leading to it or the chair becoming damaged or immobilized.

4 Technical Background

A. Navigation/Motorization - Sebastian, Aya, and Maha

The initial technical advancement in transforming a standard wheelchair into an autonomous, controllable model involved the motorization of its wheels. For our motor mounting module, we have attached our batteries, motors, servo drivers, encoders, and microcontrollers to a single stiffened piece of plywood which can be attached and detached from the wheelchair as shown in section 2.5. This allows the user to choose when their chair is motorized or manual.

After having motorized the wheels, we were then able to take on the implementations of our different navigation operating modes. For the joystick operating mode, we purchased a joystick with internal electronic speed controllers that regulate the voltage from the batteries applied to each motor according to the direction of the joystick's input and the current speed setting. For our remote navigation, we implemented 2 Accelnet Micro Panel digital servo drives which use encoder feedback from each wheel to precisely regulate the speed of its respective motor according to the PWM inputs being received from the Arduino MEGA.

To enable the diverse operating modes envisioned, the Arduino MEGA incorporates distinct functions essential for basic navigation: moving forward, reversing, and executing left and right turns. Moving forward or backward was achieved by synchronizing the speeds of the left and right motors, with the direction reversed by applying a negative speed value. Turning was more complex since we wanted to ensure the turns were sharp enough but also not too jerky; we experimented with several methods before deciding that deactivating one motor while the other remained active provided the most efficient maneuvering. Specifically, to turn right, the right motor is turned off, allowing the left motor to pivot the chair, and vice versa for turning left.

The primary electronic control of the wheelchair is managed via an NVIDIA Jetson Nano which is a small AI computer capable of processing multiple software required for the wheelchair such as object detection, wheel movement, and feedback. However, as controlling the wheelchair's motors requires six PWM pins, which the Jetson Nano does not possess, we opted to use an Arduino MEGA which offers ample pin availability. This setup enabled serial communication between the Jetson Nano and the Arduino using the PySerial Python library, facilitating wireless operation. Unlike Arduinos which require a direct connection to a computer, the Jetson Nano can communicate remotely over SSH, enhancing the wheelchair's wireless capabilities.

We further developed a Python script utilizing the Curses library to interpret key presses from a laptop remotely connected to the Jetson Nano. This script allowed for real-time control of the wheelchair using the laptop's arrow keys: the up arrow to move forward, the down arrow for reverse, and the right and left arrows for turning. Releasing a key sends a "stop" command to halt the wheelchair.

For voice-controlled operation, our attempt involved using the PocketSphinx Python library. A USB microphone attached to the Jetson Nano will capture spoken commands

such as "forward," "backward," "turn right," "turn left," and "stop." These commands will then be transmitted to the Arduino, triggering the corresponding motor actions. The script we currently have was written in such a way that the same command is continuously sent to the Arduino for execution until a new command is recognized.

Our approach underscored the project's modularity, demonstrating that the same underlying motor control code could be adapted to multiple input methods. This flexibility invites future enhancements and the integration of additional functionalities.

Although the LIDAR-based navigation mode is not yet operational, we have laid the groundwork by integrating a LIDAR sensor and configuring the Robot Operating System (ROS) to process environmental data. This setup not only tests the current capabilities of the wheelchair but also opens avenues for future teams to expand on this project. The modular design of our system encourages ongoing innovation, allowing new features and control methods to be seamlessly integrated as technology advances.

B. Robotic Arm/Object Detection- Norbert and Aya

The robotic arm is a version of the Inmoov robotic arm. Some changes have been made to alter the integrity of the material to better suit it to a functional role, not a purely artistic one.

The arm was 3D printed mainly in the ECE lab and the printers available there. There were many issues with the parts coming out in unusable states, and requiring many hours of filing and drilling to have them fit together as intended. This may be a result of poor-quality materials or errors in printer set-up. The lack of experience with the 3D printers available to us may also be a cause.

To control the arm, an Arduino code is used to control the motors in the base of the arm, directing the fingers to open and close or the wrist to twist. For lack of space on the arm, and issues with having enough torque to rotate the arm, the arm is attached to a wooden board, which would then be lifted or lowered, to simulate the vertical motion of the arm.

To build the object detection module of our wheelchair, we first built the jetson-inference module from source using the repository from https://github.com/dusty-nv/jetson-inference. One reason behind using jetson-inference was that the real-time object detection libraries compatible with the NVIDIA Jetson Nano are very limited, and the jetson-inference library was made specifically for the NVIDIA Jetson Nano. Building this library and using it therefore was a solution to lots of compatibility issues that we had encountered when attempting to achieve object detection on the Jetson Nano.

The hardware setup requires that the Raspberry Pi SC0024 IMX219 NoIR Camera be connected to the Jetson Nano via a ribbon cable. To facilitate communication between the camera and the Jetson Nano, it's essential to determine the camera's device on the Jetson Nano. In our configuration, the camera index was identified as 0.

Next, we downloaded and experimented with different models pre-trained on the COCO dataset. We concluded that the best two models to use would either be the Mobilenet-SSD dataset or the Inception-V2 dataset. While the Mobilenet-SSD was faster, the

Inception-V2 dataset was more accurate while still being fast enough (20 fps) for our application. We then incorporated the Inception-V2 model into our inference script.

5 Relevant Engineering Standards - Aya, Maha

In terms of code design for our project, we adhered to the PEP 8 coding standards for the style of our Python code to ensure the software's integrity and readability. We also ensured our code was documented and commented on to facilitate debugging and to ensure an easy pass-off to future teams who decide to build off of our project.

Communication protocols also played a critical role in the success of the project. For local control, we implemented Serial Communication standards to interface the NVIDIA Jetson Nano with the Arduino MEGA, facilitating robust and real-time command processing. For remote interactions, we utilized secure shell (SSH) protocols to ensure encrypted and secure wireless communications, critical for preventing unauthorized access and control of the motorized wheelchair. Moreover, the communication between the NVIDIA Jetson Nano and the Raspberry Pi SC0024 IMX219 NoIR Camera involved the usage of the MIPI Camera Serial Interface Protocol. These communication protocols not only provided the necessary functionality but also ensured that our system adhered to established cybersecurity practices, therefore prioritizing the user’s safety.

As for the hardware, we adhered to basic safe electrical standards such as unplugging power when not in use, and keeping ground and hot wires away from each other.

6 Cost Breakdown- Maha

The cost breakdown approximates the costs to build one MOSS Chair with the basic features that we initially specified. Our main components were the Jetson Nano, sensors, batteries, and motors. Professor Osama, the customer, did not specify any components, so we were given full flexibility in building the MOSS Chair, as long as it fulfilled the requirements specified in the Appendix. However, Professor Osama bought all of our components with some of them being donated by the team members themselves.

Project Costs for Production of Beta Version (Next Unit after Prototype)				
Item	Quantity	Description	Unit Cost	Extended Cost
1	1	NVIDIA Jetson Nano	149	149
2	1	Arduino Nano	26	26
3	2	Raspberry Pi SC0024 IMX219 NoIR Camera	49	98
4	1	LiDAR sensor	99	99
5	1	Brushed motors with gearboxes and joystick	259	259
6	2	Electronic Speed Controllers	100	200
7	1	AGM Battery	70	70
8	2	Rotary Encoders	19	38
9	2	Shaft Coupler	15	30
10	2	ADC Driver	7.50	15
11	2	Orientation Sensors	33	66
12	1	Voltage Regulator	10	10
			Beta Version-Total Cost	1,060

7 Appendices

7.1 Appendix A - Specifications - Maha, Norbert, Sebastian

Requirement	Value, range, tolerance, units
High Object Detection Accuracy	>= 90% Accuracy
Real-Time Processing	<= 500 ms processing time
Night Vision Object Detection	No change in detection accuracy under different brightness conditions
Max Speed	4.5-8 mph
Battery Life	≈8 hours, must be adjustable(modular battery)
Arm Weight	12 lbs
Load that arm can lift	5 lbs
Arm reach length	3 ft

7.2 Appendix B – Team Information

Sebastian Gangemi is a graduating Computer Engineering major with interests in connected systems and data science. After graduation, Sebastian plans on pursuing a career in software engineering.

Aya Kassem is a graduating Computer Engineering major with a passion for applying Computer Engineering and Machine Learning concepts to the Biomedical and Healthcare fields. After graduating, Aya will be pursuing a PhD in Computer Engineering where she will be researching the applications of Artificial Intelligence in Medical Imaging.

Norbert Malik is a graduating Mechanical Engineering major looking to go into a Masters in Civil and Environmental Engineering. He hopes to work on bridges, and other structures, that have moving components, using his mechanical engineering background in tandem with his Civil Masters to create such designs.

Maha Noor is a graduating Biomedical and Electrical Engineer who has a passion for biotechnology and semiconductor technology. She hopes to use her skills as an engineer to work on the electrical components present in biomedical engineering. Starting this Fall, Maha will be joining Texas Instruments as an applications engineer.