Boston University
Electrical & Computer Engineering
EC463 Senior Design Project

Final Prototype Testing Plan

MOSS Chair: Modular Open-Source Smart Wheelchair

by

Team 11
MOSS Wheelchair

Team Members

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

I. Required Materials

1. *For Motor Controller:*
   A. Hardware
      - Wheelchair
      - 2 50 W 24 V Brushed Motors
      - 2 Motor controllers
      - 2 24 V AGM Batteries
      - 2 Single-ended quadrature wheel encoders
      - 2 Single-ended to differential quadrature signal converters
      - Joystick
      - Arduino
      - NVIDIA Jetson Nano

   B. Software
      - Arduino script
        - Controls RPM of the motor through the motor controllers
      - NVIDIA Jetson Nano Script
        - Uses PySerial library for Jetson Nano - Arduino communication
        - Uses Curses Library for key listening
        - Communicates with Arduino to send commands to move forward, backward, turn right, turn left, and stop

2. *For Robotic Arm*
   A. Hardware
      - 3D printed forearm and hand parts, assembled
      - Strings to attach and move the arm and fingers
      - Adhesives and fasteners such as screws and bolts

3. *For Object Recognition Software:*
   A. Hardware:
      - NVIDIA Jetson Nano
        - Flashed microSD card
        - Power adapter (2.5 A 5 V)
        - WiFi card
        - WiFi antennas
      - Raspberry Pi SC0024 IMX219 NoIR Camera
      - Objects that you want to detect

   B. Software:
      - Python 3.6
      - Jetson Inference

- PyTorch with CUDA support
- TorchVision
- GStreamer

II. Set-Up

The equipment and setup are divided into 2 parts: The motor controller, and the robotic arm.

Joystick and Motor Controller: There are two separate controlling systems we will test across a variety of loads. The first controller is the joystick which requires 3 connections: Power, Left Motor, and Right Motor. (Power is always last to plug in and first to unplug) The second controller will be an Arduino microcontroller. In this setup, each motor drive gets connected to Power and its respective motor, and the Arduino gets plugged into the NVIDIA Jetson Nano to receive speed inputs from the serial monitor. The reason for the Jetson Nano is that, unlike the Arduino, it does not have to be directly plugged into a laptop and it can rather be controlled remotely through SSH. We however still need an Arduino because the Jetson Nano only has 2 PWM output pins, while we currently need 6 to control the wheelchair. We therefore used PySerial to make the Jetson Nano communicate user inputs to the Arduino from a remote laptop, and the Arduino then relays the commands to the rest of the hardware. The Arduino sends the speeds to each motor drive, and the motor drive regulates the speed from 4 rpm to 16 rpm for whichever load it is currently tuned for. To switch loads, as there is no current weight measurement to dynamically change the tune, you must upload the corresponding file to both motor drives one at a time.

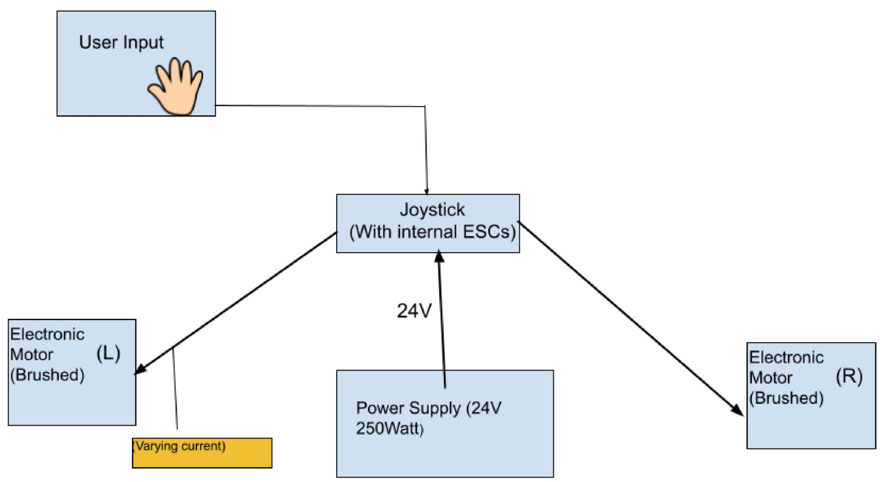
Robotic Arm:
We will have printed all of the arm parts and have assembled the arm. To force the motion of the finger, we have lines attached to them all to show the range of motion of the fingers. Additionally, we will be demonstrating the mechanical soundness of the printed parts. The parts are expected to show no distress such as cracks or bends. We will also have the connected parts be pulled on to show that the long connections are sturdy.

Object Recognition: The Jetson Inference library that we are using already contains a pre-trained Inception-V2 model trained on Microsoft's COCO dataset, which contains multiple instances of common objects. Today, we will import this model and use it to detect objects in real time from the Raspberry Pi SC0024 IMX219 NoIR Camera connected to the Jetson Nano and classify whether these objects were detected correctly or not.

III. Pre-testing Setup Procedure

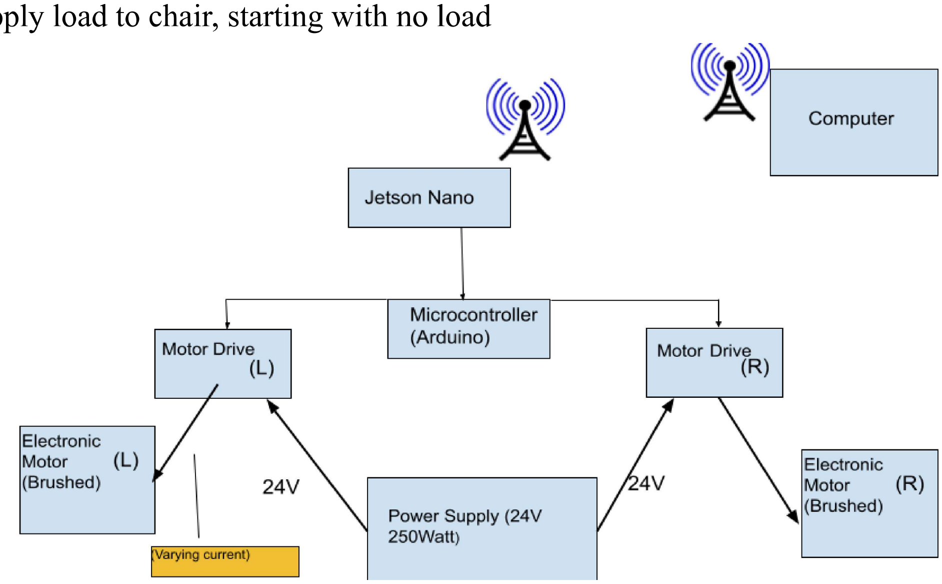
Joystick:

1. Plug in the joystick's 3 connections(Power, Left Motor, and Right Motor).
2. Establish a navigation course to test controllability.



Motor controller:

1. Plug in motor controller’s 4 connections(Power, Left Motor, and Right Motor, Arduino).
3. Upload control_chair.ino script to Arduino
4. SSH into the Jetson Nano
5. Activate the virtual environment containing the required libraries
6. Navigate to the directory containing the control_chair.py script
7. Run the control_chair.py script using the command: sudo *python3 control_chair_keys.py*
8. Control chair by holding down up, down, right and left keys
9. Apply load to chair, starting with no load



Robotic Arm:

1. Inspect the fingers and arm to make sure it is holding together
2. Inspect the strings to ensure they are all unobstructed and not damaged
3. Ensure the parts are well attached
4. Pull strings to show the hand gripping and releasing
5. Demonstrate wrist rotation

Object Recognition:

1. Make sure you have a version of Python 3 (preferably python 3.6 to avoid dependency issues) installed on your Jetson Nano
2. Build the jetson-inference module from source by cloning the following repository and following the instructions listed on it: https://github.com/dusty-nv/jetson-inference
3. Make sure your Raspberry Pi SC0024 IMX219 NoIR Camera is connected to your NVIDIA Jetson Nano via a ribbon cable
4. Find the index of your camera by running the following command on your Jetson Nano: *v4l2-ctl --list-devices* . In our case, the index of our camera is “0”
5. Download the pretrained InceptionV2 model
6. Run the script with the command: *python3 test_inference.py*

IV. Testing Procedure

Joystick:

1. Pick a starting and ending point for the chair
2. Navigate the chair from start to end point using joystick directional input
3. Repete with higher speed levels

Motor Controller:

1. Enable motors
2. Pick a starting and ending point for the chair
3. Navigate the chair from start to end point using laptop’s up, down, right and left arrow keys

Robotic Arm:

1. Secure arm to table to ensure it does not move with hand off edge
2. Move at the hinge joints using attached strings to test the angle that can be achieved with the printed parts
3. Hold an object while keeping fingers in tension
4. Release object by relaxing fingers
5. Hold object up while keeping fingers outstretched
6. Release object by relaxing fingers

Object Recognition:

1. Power on the Jetson Nano
2. Navigate to the directory that contains the test_inference.*py* Python script and the pre-trained Inception-V2 model
3. Run the script: *python3 test_inference.py*
4. Wait for the screen to show your real-time camera recordings
5. Place objects to detect in front of the webcam
6. Check whether there is a bounding box around the object to determine whether the model recognizes an object is present in the frame
7. If a bounding box exists, check the label of the bounding box to determine whether the object has been classified correctly

Object	*Detected Object in Frame?*	*Classified Object Correctly?*

V. Measurable Criteria

Joystick:

1. User is able to reach end destination
2. Different magnitudes of motor speeds are displayed as user changes speed settings

Motor Controller:

1. Arduino control system should be able to regulate speeds of wheels as low as 4 rpm and as high as 16 rpm
2. Arduino control system should be able to move forward and backward with up to 120 lbs load applied
3. When controlled remotely from laptop, the chair should be able to navigate with the up, down, right and left keys through any realistic route
4. When controlled remotely from laptop, the chair should be able to quickly respond to key presses and key releases

Robotic Arm

1. Fingers can all close independently and together
2. Hand can support weight of an object when gripped and outstretched
3. Parts are still mechanically sound: no cracks, breaks, or bends when loaded
4. Motor control is decent

Object Detection:

1. The model should detect that there is an object in the frame at least 80% of the time
2. The model should correctly classify the object at least 60% of the time
3. The model should detect the object’s presence within a second