Augmented Mobility Platform Education (AmpEd)
Daniel Beckett, Khaled Khaled, Lauren May, Taylor Yee

Project:
Augmented Powered Mobility

Design Review 4 and User Manual

Emails of Team Members:

Kaa325@nau.edu – **Khaled Khaled**

Dbb72@nau.edu – **Daniel Beckett**

Lm933@nau.edu – **Lauren May**

Tey24@nau.edu – **Taylor Yee**

Client & GTA emails:

Kyle.Winfree@nau.edu – **Dr. Winfree**

lz239@nau.edu – **Liming Zheng**

Faculty Advisors:
Client: Krista Branch, PT, PCS, ATP
Sponsor: Dr. Kyle Winfree
Faculty Mentor: Liming Zheng

TABLE OF CONTENTS

Introduction to Design Review 4

Welcome

As senior-level electrical engineering students at Northern Arizona University (NAU), we have been tasked with building a project that would fulfill the needs of a client residing in Flagstaff. Examples of clients' needs include a grid-connected T-type power converter, an automated orchidarium, and a cave climate monitor, to name a few. Each team of students was assigned to a client by Dr. Winfree, with careful consideration to the skills that each student possessed. Our team was formed with diversity in mind: we're a combination of communications, power, embedded systems, and software engineers in the making.

Our client, Krista Branch, is a physical therapist, pediatric certified specialist, and assistive technology professional, who works with disabled and special-needs children in the Flagstaff Unified School District (FUSD). She has previously worked with a team of engineering students at NAU to develop the Augmented Powered Mobility (APM) platform, but currently needs a robust, adaptable, and updated platform in order to meet the needs of her students.

We are pleased that you have chosen us, AmpEd, for your business needs. There is a strong need for our augmented mobility platform, as evidenced by students who have yet to experience independent mobility: a milestone that is important to human growth and development. We provide for you here a powerful system that will allow those students to move with more freedom than they have before, and collect data to track their progress along the way. This platform has been designed with your specific needs in mind, and some of the key highlights include:

- A user-configurable joystick and head switches
- A client-side graphical user interface (GUI) that allows you to adjust data collection parameters and driving inputs
- Full driving capabilities: forward and reverse motion, as well as smooth turning
- An emergency stop switch in case the platform needs to be powered down

The Problem: Lauren May

The team's faculty sponsor, Dr. Kyle Winfree, has devoted much of his research to measuring and improving healthcare through wearable technologies. Go Baby Go! (GBG) is an example of one of his projects and provides the basis for the team's APM project. GBG is a non-profit organization that operates alongside the Cerebral Palsy Foundation (CPF). The GBG program itself started at the University of Delaware (UD), but now operates nationwide, and even has remote sites across the world. Winfree's GBG project is based out of Flagstaff, AZ, and like many other across the U.S., is aimed at providing modified toy car rides to children with disabilities. These specially modified cars are primarily geared towards young children. However, there are still other, older children who have never experienced independent movement, and are currently too big for the GBG cars. This is where the AmpEd team at Northern Arizona University (NAU) comes in.

Team AmpEd's project's goal is to design and develop electronics and data collection software to meet the needs of two different parties. First, to help the disabled children that Branch works with, to experience independent mobility and practice driving a mobility platform. And second, to provide data collection in an easy-to-read manner for the client, so that she can tailor future therapy sessions with her patients to better meet their specific needs.

Custom electric powered wheelchairs (EPWs) for children with disabilities are expensive, and in most cases, the children who need them have never experienced driving a powered wheelchair before. Currently, the process of obtaining such a device involves the student passing an exam that determines their ability to maneuver a device that they have little to no experience with. For children with major disabilities this can be a major hurdle in pursuing their own independent mobility.

Assistive devices and platforms have been made to address some of these problems, but they are also expensive, and larger in scope than what the client is seeking. The primary objective of the team is to develop a low-cost, adaptable platform that contains additional inputs to address the needs of disabled children who need powered wheelchairs.

The Design Process: Taylor Yee

Introduction and Subsystems

Our project's overarching goal was to build an adaptable platform to help students experience independent mobility, while simultaneously collecting data for our client to provide information on how the students are progressing. We chose to segment our project into three smaller subsystems: mobility, power, and the GUI. These were divided as such as we felt that they were aspects of our project that were essential to its full functionality - if one of these did not work, our project would not be successful. Furthermore, these divisions allowed us to break the project into smaller, more manageable tasks that made sense in the scope of our team's abilities. So how do our subsystems tie together? They rely on each other to work fully and create a cohesive platform. Our project needed to be able to power the rest of the system before it could move around with different user inputs. To that end, the platform needed to be able to move before it could begin to send meaningful data to the host PC to be stored for our client.

Prototypes

We chose to build three prototypes that would simulate not only the different subsystems, but the necessary connections between them. For our first prototype, we connected two DC motors to an L298D motor driver, which was then connected to an Arduino Uno and analog thumb joystick, with accompanying code. This prototype allowed us to simulate providing sufficient power to several components, and mapping analog values to drive motors forward, backward, and turn them as well. Ultimately, this helped us visualize how the mobility and power subsystems would connect. The prototype's circuit and L298D IC pinout can be seen in Figures 1 and 2, below.

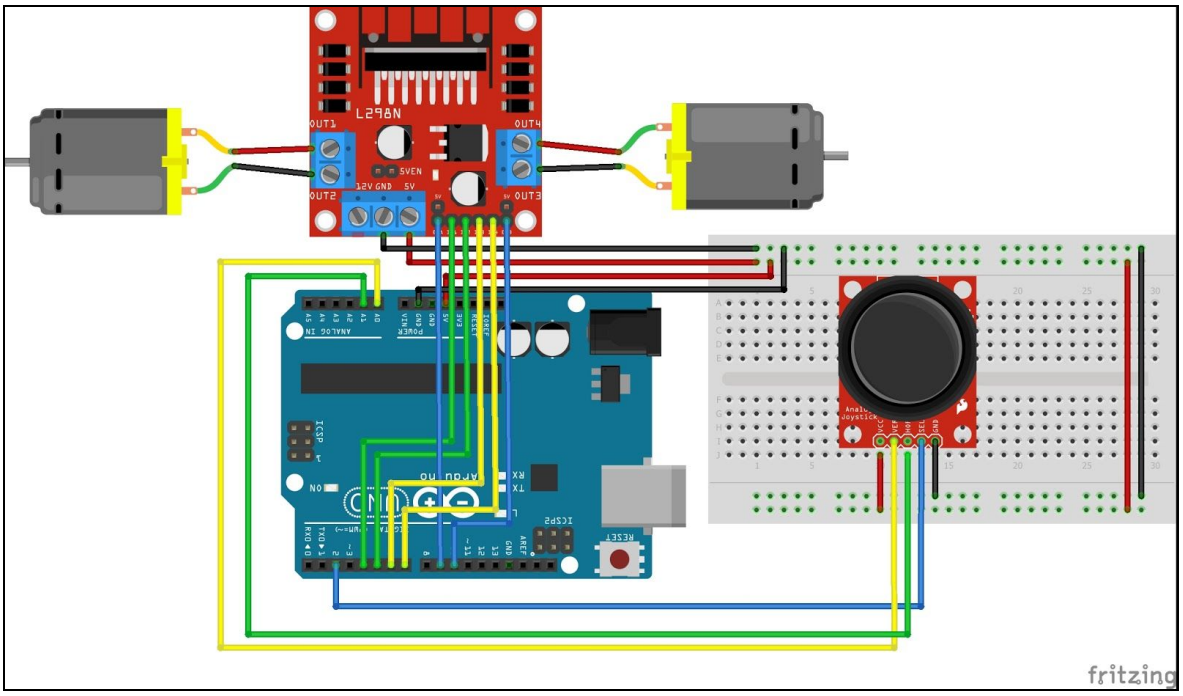

Figure 1: Prototype 1 circuit

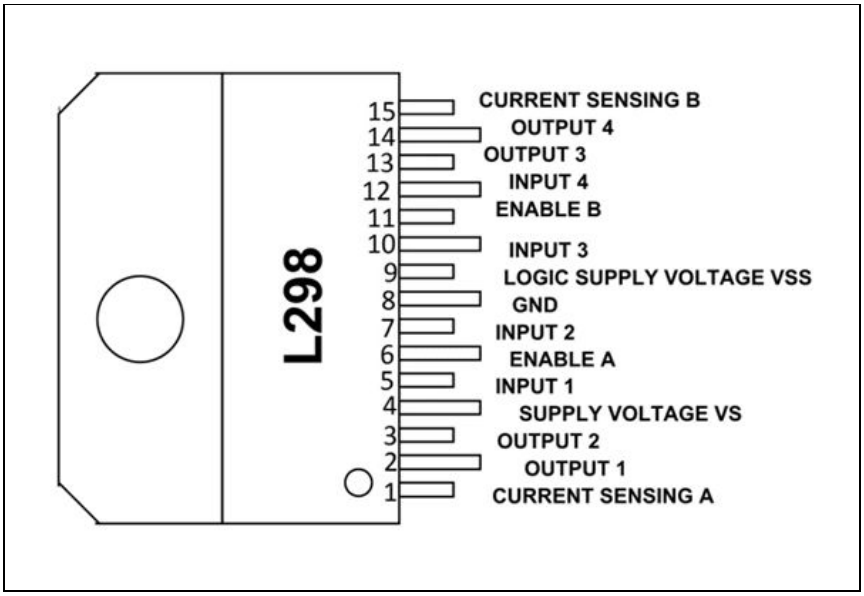

Figure 2: L298D motor driver IC pinout

Our second prototype consisted of an infrared (IR) sensor, connected to an Arduino Uno, with accompanying code. While this prototype was simpler in physical complexity, it incorporated two types of code - Arduino and Processing. The Arduino code helped us envision how we could collect data from an onboard sensor, and the Processing code allowed us to practice accessing that data in real-time and display it to the user. Furthermore, the prototype gave us a better idea of how the mobility subsystem would connect to the GUI. The prototype's circuit and a sample graph can be seen in Figures 3 and 4, below.

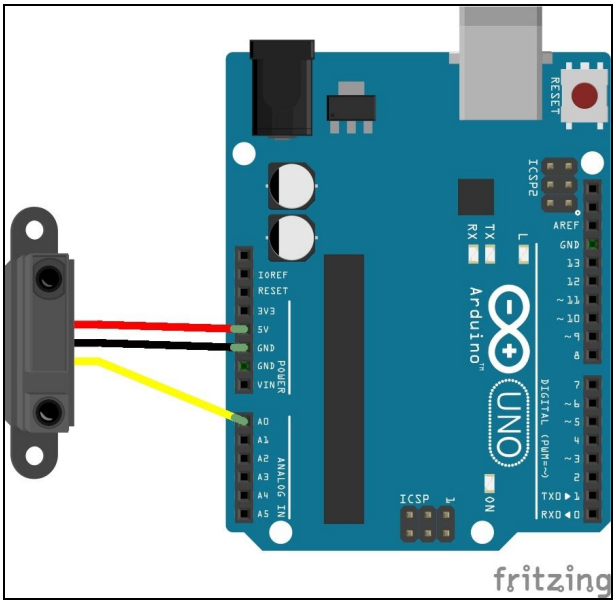

Figure 3: Prototype 2 circuit

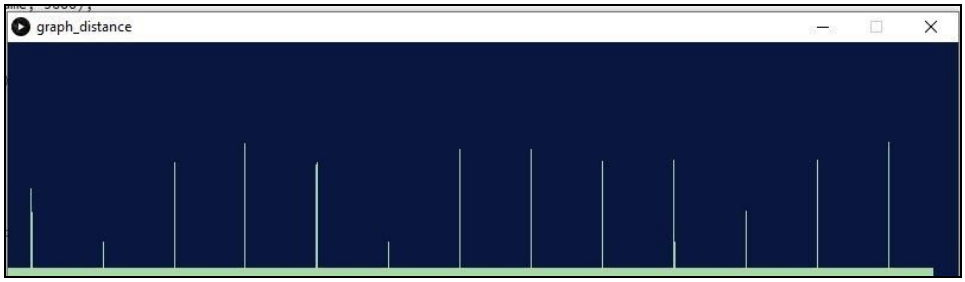

Figure 4: Distance graph with data from IR sensor

Our third prototype was not a physical build, of sorts, but rather a demonstration of the research and knowledge behind our GUI. We sketched the interface out on paper, and talked through some of its possible functionality with our client. We also established what program we would use to build our interface, and showed that we could write code for a user-interactive window that would change outputs. A circuit was built with a DC motor and four LEDs, and Processing code was written that could change motor speed and LED lighting based on a mouse click in an area of the window. With this prototype, we communicated our vision of the GUI to the client, and also established a way to communicate with the rest of the platform from the host PC. The prototype's circuit and interactive window can be seen in Figures 5 and 6, below.

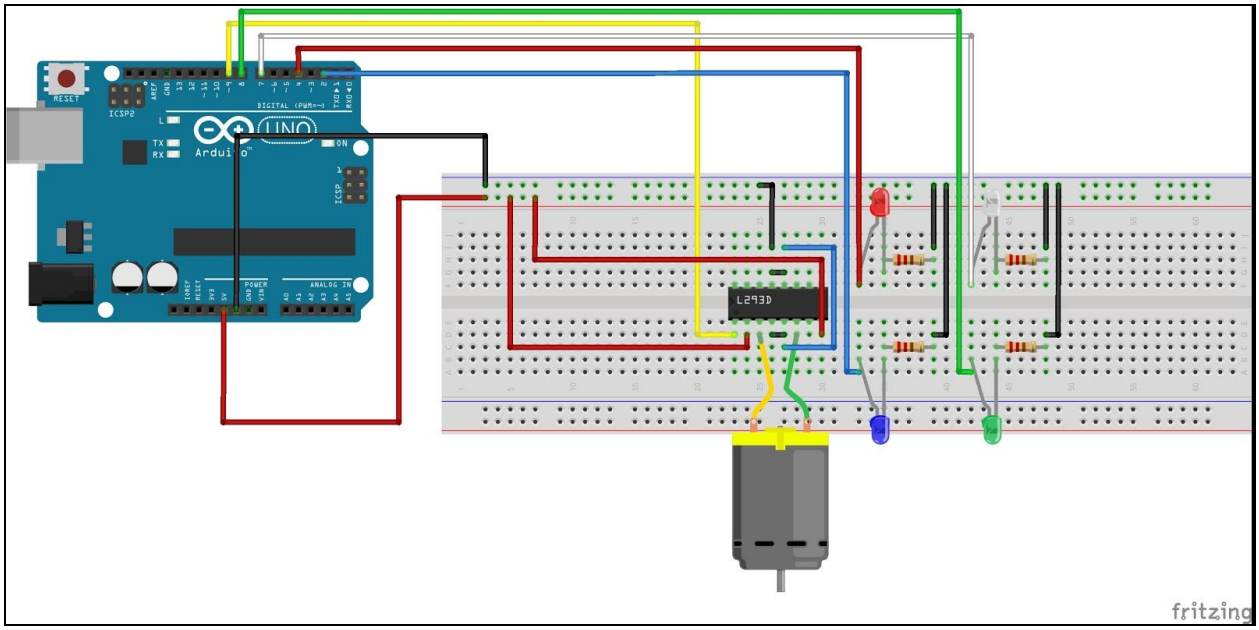

Figure 5: Prototype 3 circuit

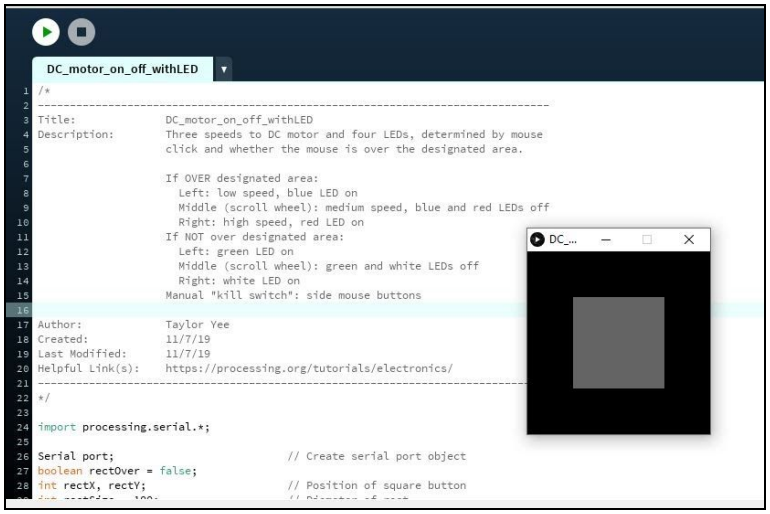

Figure 6: Interactive window and control scheme

The Constraints: Lauren May

The main target for our project is children with severe disabilities which means there are certain requirements that should be met in order to give each child an experience customized to their needs. The client was specifically looking for the following: various forms of driving, configurable inputs, and PC side GUI to adjust parameters and data collection. Since the children our client works with have a multitude of severe disabilities, there must be at least two different ways of controlling the platform, one of them being a joystick. This will help accommodate those with impaired hand-eye coordination abilities and allow them to use an alternative that better uses their strengths. To ensure that multiple controls can be used, we also built our project around adaptability. This way, if our client had a new student who had certain needs which could not be met by the current specifications (a joystick and head switches), the platform could be updated with new inputs. Another key component from the client was that the input controls, mentioned above, need to be configurable. We were able to accomplish this by having different settings for each input that allow the client to adjust the controls. Our client also specified that the final product must have a PC side GUI to adjust parameters and data collection. To accomplish this, we used a program called Processing to create an interface that was user-friendly and easy to understand.

There were other constraints that came about during the design process due to budget, parts, or time. Many of these constraints are a result of our $500 budget, just as the choice on sensors came about. We originally had an idea to include a LIDAR sensor which has the ability to map the surrounding areas and would give us access to more data as a result. This became unrealistic due to the complexity but mainly due to the low remainder on our budget. This is also the reason for the lack of a full-sized joystick included in the final result. Time was also a major constraint

since purchasing parts caused delays, so the design had to be simplified and a PCB was not able to be ordered.

Status of Planned Features (WBS)

Discussion of WBS and Gantt Chart: Taylor Yee

Our original Work Breakdown Structures (WBS) were fairly ambitious, but gave us a tangible roadmap to work with. They can be seen in the following three figures.

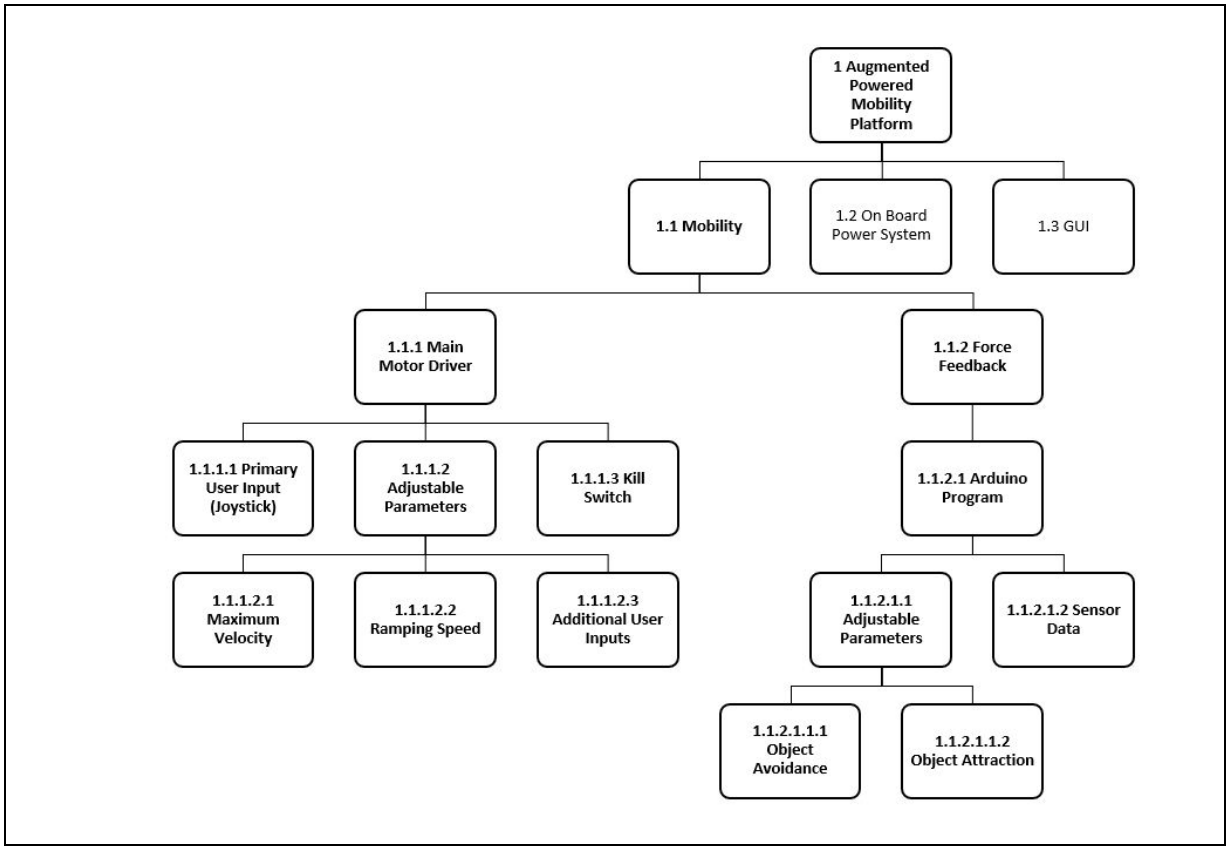

Figure 7: Detailed mobility subsystem

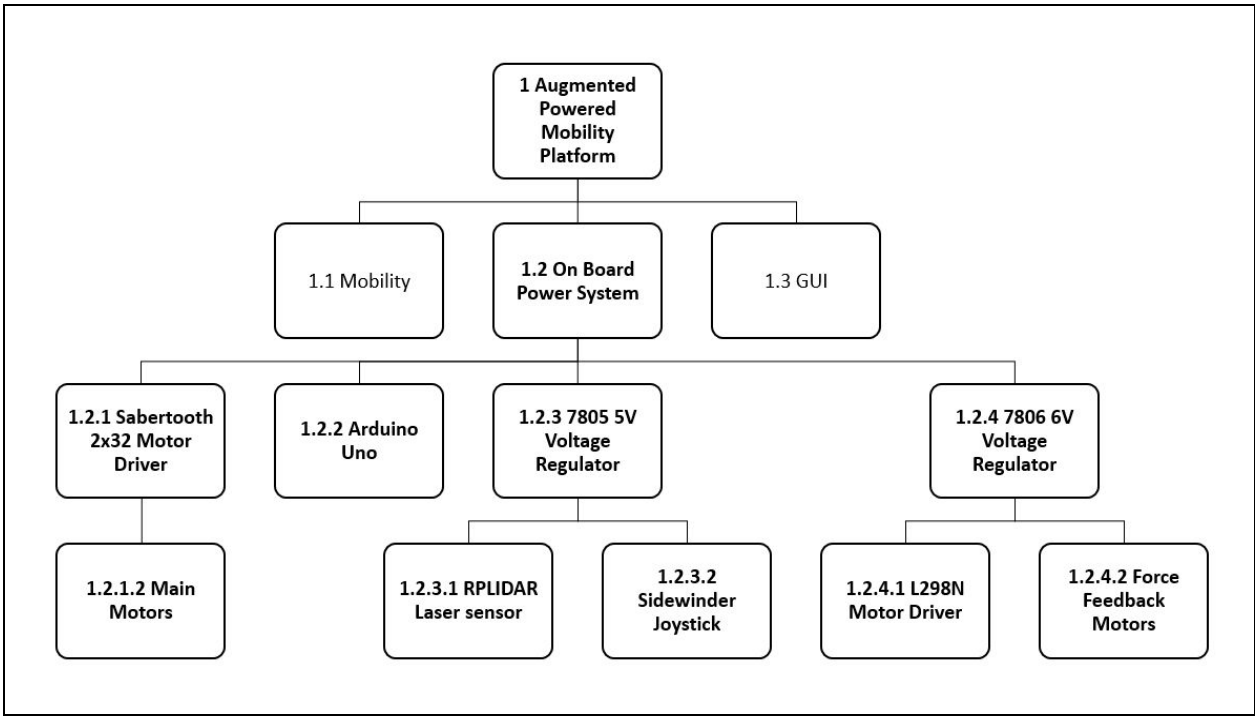

Figure 8: Detailed power subsystem

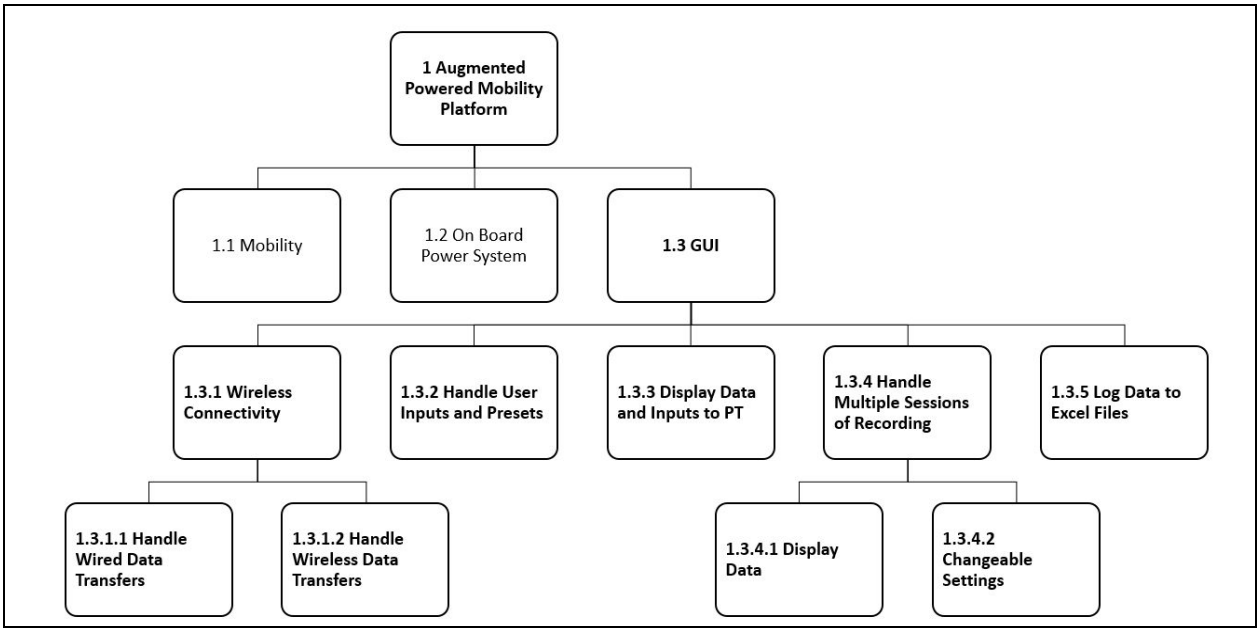

Figure 9: Detailed GUI subsystem

While we wanted to include all of these aspects into the final product, our Gantt chart below depicts what was actually completed, and when it was done.

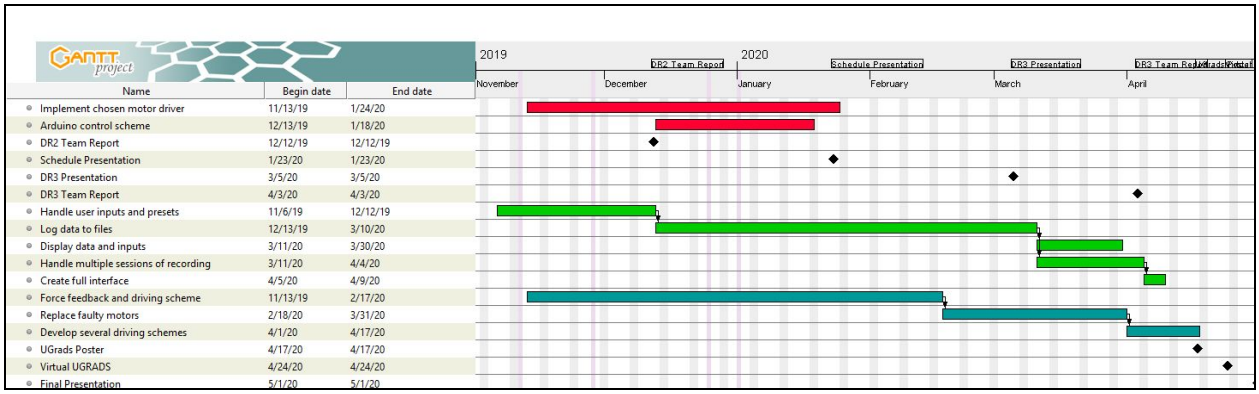

Figure 10: Final Gantt Chart for the project

Mobility Subsystem Changes

We were able to implement the main motor driver and all accompanying features, which are discussed in the accompanying updates below. However, due to the client's changing needs, we no longer needed to implement force feedback. As a result of this, several aspects of the power subsystem changed.

Power Subsystem Changes

We needed to upgrade the Sabertooth 2x32 motor driver to the more powerful 2x60 motor driver in order to supply the platform's high power demands. As a result of this, however, we were able to power the main motors, the Arduino Uno, the sensors, and the host PC. We chose to eliminate the RPLIDAR laser sensor, as it proved to be far more complex and overachieving than what our client needed. As a result of this, as well as the scrapping of the force feedback aspect, we no longer needed either voltage regulator.

GUI Subsystem Changes

Wireless connectivity was considered one of our lower priorities, as we wanted to ensure wired communications before attempting to transition to wireless. As a result, due to time constraints and its lower level of importance, it was not implemented into the final product. Furthermore, when we met with the client, it became clear that she was only looking for a few data points, and not a few hundred. As a result, we could pare down to a text file instead of an Excel file. The rest of the aspects of this subsystem remained unchanged.

Mobility Subsystem Updates

Be able to drive the platform with several user-configurable inputs

As demonstrated in our DR4 presentation, we are able to drive the platform in a typical way with a joystick, including forward and backward movement, as well as pivoting to turn left or right. Although it was not shown in the video, we created an alternative scheme: pushing the joystick in any direction results in forward movement, and clicking head switches allows the user to turn.

A challenge that we faced was being able to switch configurations on the fly without having to restart the platform, change the wiring, or change the code. To overcome this, we created a control scheme in the GUI that could choose a driving option in the Arduino code, which would then direct how the motors would operate. We also consolidated the unidirectional joystick and head switch schemes, as they resulted in identical operation. Finally, we tested each driving scheme individually, and simulated the switching modes with LEDs, to ensure the correct signals were sent.

Create a smooth driving experience for the user

Our platform is able to provide a smooth ride for the user, though this could be improved. The initial acceleration and deceleration of the platform are a bit abrupt. However, both driving in a straight line and pivoting to turn are smooth motions, as evidenced by our demonstration in the DR4 presentation. One of the biggest challenges we faced was testing the driving schemes that we implemented, due to difficulties with the previous team's motors. To overcome this, we consulted our prototype driving scheme, as we ended up using the same joystick for both the prototype and the final product. Because of this, we only needed to test a different mode with the joystick, and adapt our plan to fit the style of the head switches.

Sustain a drive time of at least half an hour

Regarding this goal, we are about halfway there. During our last set of testing, we determined that the platform could sustain about 15 minutes of continuous driving. This can primarily be attributed to us focusing on the fact that in early testing, the platform would overheat after a very short period of time. Discovering the source of this issue took time, and we could not make significant progress on the mobility of the platform until this was fixed. Once we narrowed it down to the motors causing overheating, we had to order new parts, which further delayed this. We also noted from the previous capstone team's documentation that the batteries used on our platform tend to have a lifespan of about 2-3 years, or 500-1000 charges. Since we had no knowledge of the usage history of their platform, we had a difficult time pinpointing why the batteries could not sustain the platform for long. Consultations with Dr. Winfree also led to dead ends, as he also could not figure out the issue. Buying new batteries would have put us massively over budget, so we had to settle for the drive time we could get.

On-Board Power Subsystem Updates

Completely on-board, no external sources

Our power system is capable of powering the onboard sensors, Arduino microcontroller, motors, and host PC, without overheating or undersupplying any of the parts. One of the biggest challenges here was providing enough energy to sustain motor operation for more than a few minutes at a time, as the platform would overheat and had high power requirements. To remedy

this, we isolated parts of the platform until it was discovered that the previous team's motors were worn out, most likely due to their pulse-width modulation (PWM) mode of operation. We then purchased new motors and thicker gauge wire to better support the large power draw.

A readily accessible emergency kill switch to halt operation

This was one of the first features that we wanted to focus on, as the driver's safety was an important priority for the team. As a result, this aspect was finished back in December 2019. The switch works by twisting it to turn on the platform, and pushing the button down to turn it off. The biggest challenge to this was that the platform would overheat rapidly, within moments of powering on the device. However, it was eventually determined that the older motors, and not the switch, were causing the platform to overheat. To overcome this issue, new motors were purchased, and overheating has not been a problem since then.

A simple but efficient charging method

This feature is complete, but can definitely be improved upon. Our current implementation uses two 12V batteries. We can charge them with the provided 12V battery charger, but this means that we have to charge them separately. However, the process to charge them is simple, as we only have to plug in the charger to a wall outlet. The biggest problem that we encountered was the lengthy time required to fully charge them. While this is not detrimental to the overall functionality of our product, purchasing a 24V charger would allow us to charge both batteries simultaneously, thus shortening the amount of time required.

GUI Subsystem Updates

Automatic wireless establishment of host PC with platform

As it stands, our GUI cannot automatically establish a wireless connection from the host PC to the platform. We did some preliminary research on ways to implement the wireless connectivity, looking at both WiFi and Bluetooth possibilities. There is room on the platform for a Bluetooth module to be connected, and with more time, this could most likely be implemented. However, establishing a wired, consistent connection from the PC to Arduino took longer than anticipated. Furthermore, we encountered several difficulties with choosing a suitable Bluetooth module, as well as establishing any communication via Bluetooth. With these hurdles in mind, we chose to focus on the more essential aspects of our platform (sufficient power, movement, some user adjustability from the PC), and did not have time to circle back to the wireless aspect as we intended to.

User PC side GUI to adjust parameters and assess driving skills

Our GUI is able to send command signals to the Arduino on the platform to choose the driving mode for the student, and thus adjust parameters. In future work, we could also implement adjustability for top motor speed, and other needs as the client determines them. Our GUI can

also assess driving skills, as we are able to use on-board sensor data to calculate the number of times the platform collides with an object, and the total drive time. One of our greatest challenges here was determining a control scheme that could work with both directions of data transmission simultaneously - being able to send signals to the Arduino, receive data from it, and only collect data when needed (i.e., the end of a driving session) to prevent excessive transfers. To overcome this, we chose to send different characters from the PC to the Arduino (and vice versa) to denote different driving modes (e.g., 'H' for "head switches"), and use delimiting characters that would mark the end of a data transfer, so that we only collected what we wanted.

A way to capture and log information

Our platform is able to capture and locally display and save data that the client has requested. This includes metrics such as the student's name, the date and time of the session, the total drive time, the number of platform collisions, and the type of driving mode chosen. These parameters are then saved to a text file on the host PC, in the same folder as the GUI code, with the name of the student as the file name. One of the greatest challenges we encountered was in choosing a way to store the data. Our original design required storing several hundred data points per session, so we had planned to use a microSD card and data logging shield with a real-time clock. With this method, we encountered difficulties in getting a reliable and consistent timestamp, as well as collecting data accurately - sometimes the output would not be in plain English. However, the decision to switch to a text file saved locally, with only a few lines total, eliminated these issues.

Introduction to User Manual

Welcome

We are pleased that you have chosen AmpEd for your business needs. There is a strong need for an augmented mobility platform, as there are many children with severe disabilities who would benefit from independent mobility. We provide for you here a powerful system for independent mobility that is adaptable to meet any need. This platform has been designed with your specific needs in mind, and some of the key highlights include:

- A user-configurable joystick and head switches
- A client-side graphical user interface (GUI) that allows you to adjust data collection parameters and driving inputs
- Full driving capabilities: forward and reverse motion, as well as smooth turning
- An emergency stop switch in case the platform needs to be powered down

The purpose of this user manual is to help you, the client, successfully use and maintain the augmented mobility platform in your actual business context going forward. Our aim is to make sure that you are able to benefit from our product for many years to come!

Team AmpEd's project's goal is to design and develop electronics and data collection software to meet the needs of two different parties. First, to help the disabled children to experience independent mobility and practice driving a mobility platform. And second, to provide data collection in an easy-to-read manner for the client, so that they can tailor future therapy sessions with her patients to better meet their specific needs.
Custom electric powered wheelchairs (EPWs) for children with disabilities are expensive, and in most cases, the children who need them have never experienced driving a powered wheelchair before. Currently, the process of obtaining such a device involves the student passing an exam that determines their ability to maneuver a device that they have little to no experience with. For children with major disabilities this can be a major hurdle in pursuing their own independent mobility. Assistive devices and platforms have been made to address some of these problems, but have a limited scope of what they can accomplish. Our product is a low-cost, adaptable platform that contains additional inputs to address the needs of disabled children who need powered wheelchairs, while simultaneously collecting data for our clients to provide information on how the students are progressing.

The chair is separated into three subsystems: mobility, power, and the GUI. These three sections each describe an integral part of the product and tie the different functions together.

Mobility Subsystem

The mobility subsystem focuses on making sure the platform itself can “talk” to, and move in conjunction with, the electric powered wheelchair (EPW). The patient’s current wheelchair is used to expand mobility to those with severe disabilities whom traditional powered wheelchairs pose a danger in the event of an accident during the transfer from their wheelchair to the powered wheelchair. It allows the patient to use their own wheelchair while they gain experience and confidence in driving. There are adjustable parameters and different control mechanisms that can be used to accommodate a variety of disabilities that the patient might have. This subsystem is run by an Arduino Uno, which can send control signals to each component, and handle resulting data outputs accordingly.

On-Board Power Subsystem

The on-board power subsystem is devoted to ensuring that the team’s new platform can run off of two 12V batteries. This circuitry makes the chair completely self-sufficient and is easily adaptable to allow additional components to be added. Important parts include the motors, Arduino, joysticks, and switches. This also includes routing and stepping up and down the power appropriately to ensure that parts are not electrically overdriven or undersupplied, which is critical for the low powered devices.

GUI Subsystem

The GUI subsystem is responsible for displaying requested real-time data to the physical therapist in a way that is easy to read. Furthermore, the GUI handles data logging and user presets that have the ability to be changed and reset from patient to patient. This is primarily done through the included Processing interface.

Installation

Introduction and Operation Considerations

This platform is not meant to be operated in rainy, snowy, or wet conditions. The ride itself is smooth and turns well, but it is not recommended to drive it through narrow areas, as the turns have a wide radius.

Charging the Platform

To charge the batteries, first make sure that the kill switch has been pressed. Then disconnect any wires connected to their terminals, the order in which you connect or disconnect doesn't matter. (It is recommended that after disconnecting a terminal, to keep its specific loose wires together by re-inserting the bolt from their terminal.) Once the batteries have been disconnected, charge them separately using the 12V car battery charger. Both batteries must be charged completely to 12V. To ensure a longer operation life, check that both battery voltages are within 0.1V of each other before re-installing.
To re-install the batteries, make sure that the kill switch has been pressed. Then attach each terminal's corresponding wires in any particular order.

Hardware Mounting To Be Done

Most of the hardware has already been installed and mounted for you, and the platform is nearly ready to go. However, there are a couple of physical installations that still need to be done to ensure a smoother and more adaptable experience for users.

Host PC Mount

In our DR4 presentation, it can be seen that the host PC rests on the rails that connect the motor portion to the chair's platform. While this works with a plastic chair and a fully able-bodied user, this most likely will not work with an actual electric powered wheelchair. To that end, a structure to hold the host PC needs to be built for a laptop to rest on, as we do not have wireless capabilities installed.

User Input Mounts

The platform currently does not have a way to reliably mount the thumb joystick and head switches that drive the platform. As a result, they rest on the user's lap while driving the platform. The previous implementation used PVC pipes to mount the necessary inputs, and this is certainly an option to consider.

Full Size Joystick

While the platform currently uses a thumb joystick as a possible input, we recognize and understand that users would benefit from a full-size joystick that can be pushed by the hand,

without great motor control. With this in mind, implementing a new joystick would require some desoldering and resoldering of wires to ensure proper connection to the Arduino. Some Arduino setup and operation code would also need to be altered to account for these new pin connections.

Software

To complete your installation and begin using the platform, you'll need to install some software on the PC you plan on using with the platform.
If you've already installed the software needed, you can skip steps 1-3.

General Installations

1. You'll need to install Arduino and Processing, two code compilers, called IDEs, that are needed for proper operation. They can be found by following the two links provided:
   https://www.arduino.cc/en/main/software
   https://processing.org/download/
2. The Arduino link should provide a **.exe** file that can directly open the Arduino compiler. The Processing link will download a **.zip** file that contains the **.exe** file to open the Processing compiler.
3. Now that you've got both compilers, you'll need to import a couple of code libraries and load up the code to run the platform!

Arduino Setup

4. We'll start with the Arduino. Use the **.exe** file to open up the IDE.
5. Once that's done, download the Arduino .**zip** file from the team's website (found on the cover of this manual). Unzip the contents into a folder that you can easily find.
6. Back in the Arduino IDE, open up the **.ino** file in the unzipped folder by going to **File >> Open**, and locating the **AmpEd_Arduino.ino** file.
7. Next, go to **Sketch >> Include Library >> Add .ZIP Library…**, and select the two **.zip** files located in the folder you downloaded from the team site.
8. Next, go to **Sketch >> Include Library >> Manage Libraries**. You'll need to add one more external library, but not from a **.zip** file this time. Type "LSM9DS1 sparkfun" into the search bar, select the library and install it.
9. Click the green checkmark at the top left of the screen to "verify" your code - this makes sure that there are no errors. Once that's done, connect your PC to the Arduino microcontroller with a USB-A to USB-C cable (not included), and click the right-facing arrow to upload your code.

Processing and GUI Setup

10. Now we'll move into the Processing setup. Make sure you've unzipped the Processing files into a regular folder, then use the **.exe** file to open up the IDE. Your window should look pretty similar to the Arduino IDE window.
11. Once that's done, download the Processing .**zip** file from the team's website (same URL as step 4), and unzip the contents into a folder that you can easily find.
12. Back in the Processing IDE, open up the **.pde** file in the unzipped folder by going to **File >> Open**, and locating the **AmpEd_GUI.pde** file.
13. As with the Arduino code, we need to import one code library for the interface to work. Go to **Sketch >> Import Library... >> Add Library…** under the "Libraries", type "controlP5". You should see a library with the author "Andreas Schlegel". Click the "Install" button in the lower right hand corner of the window.

Configuration and Use

Power

To turn on the device, there is a red button mounted on the casing where all of the circuitry lies. The button acts as an on switch as well as the emergency off switch.

To turn on: turn the button in the direction of the arrows.

To turn off: press down on the button until it clicks.

Data Logging

This product is capable of logging relevant data from each driving session. This includes the student's and technician's names, the total drive time, and the amount of times a patient has collided with an obstacle. The platform is able to function without logging any data from a session; however, if you would like to use the data logging features, they are detailed below.

Starting a New Session

If the same student uses the platform for multiple sessions, the previous session's data will be overwritten with the creation of a new file. If this is not desired, see the Accessing Data From a Previous Session section to move that file before beginning to record a new session.

Open the file **AmpEd_GUI.pde**, and click the "Run" button in the top left corner of the screen (looks like a "Play" button. You should see a window pop up like the one below. A helpful instructions box will be displayed on the left side of the window, which can be followed to successfully record data and select a desired control. It is important to note that the host PC must maintain a physical, wired connection with the onboard Arduino microcontroller at all times for

a successful data recording; additionally, do not leave the student's name blank, as the session will not be recorded if this occurs.

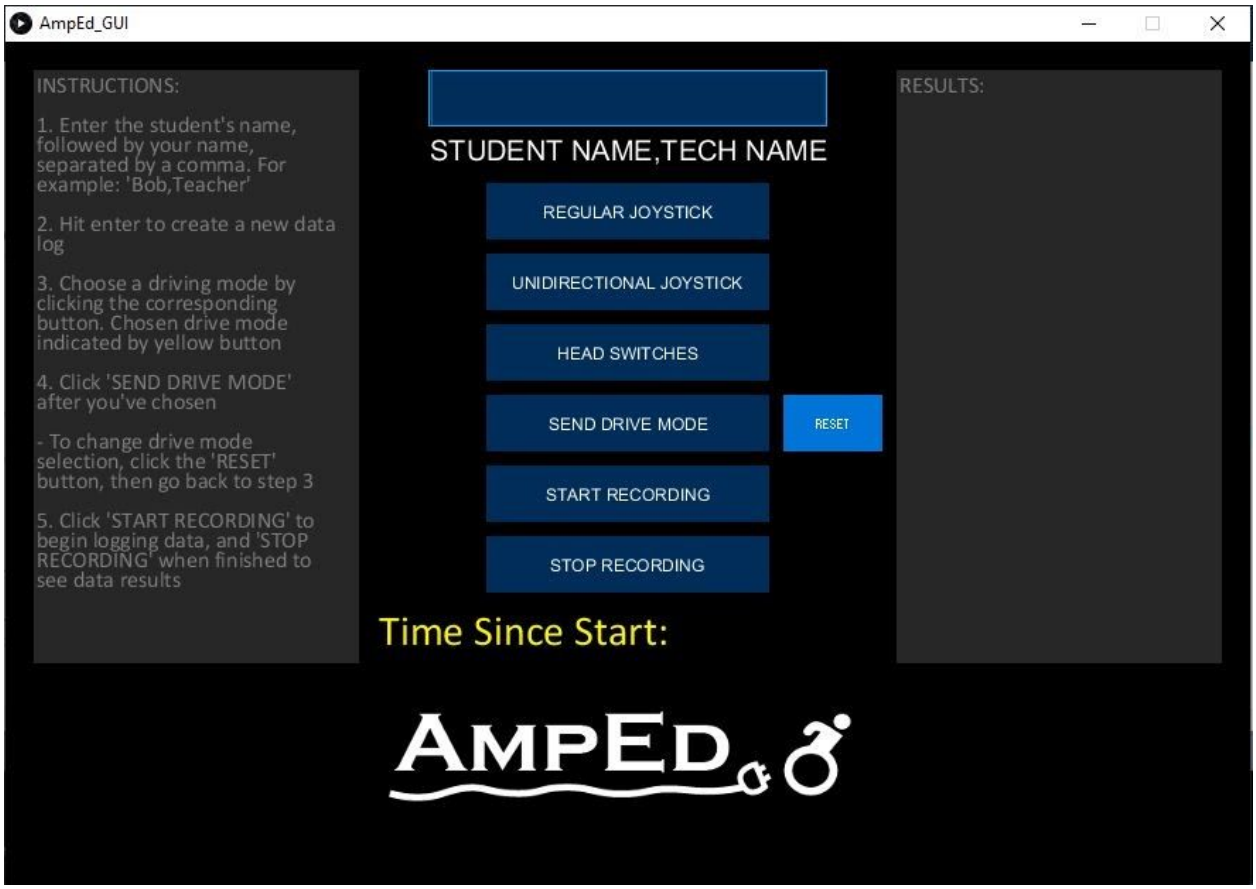

Figure 11: GUI window

Accessing Data From a Previous Session

After a session has finished, data is automatically saved to a **.txt** file, with the student's name as the filename. This file will be located in the **AmpEd_GUI** folder that was downloaded in the If the same student uses the platform for multiple sessions, the previous session's data will be overwritten with the creation of a new file. If this is not desired, see the Accessing Data From a Previous Session section to move that file before beginning to record a new session.. If you cannot find this location, you can always go to **Sketch >> Show Sketch Folder** to locate it. Once a session has finished, you can edit the file and make notes and changes as needed.

Change Settings of a Current Session

If there needs to be a setting change during a session, the session will have to reset to choose a new input. Using the main computer program, click the **RESET** button towards the right of the

screen to clear all current settings. You will not have to re-enter the student and technician name to ensure the data saves properly. From here you can pick the new settings and press **SEND**.

Connect Student's Wheelchair

Attaching and detaching a push or electric powered wheelchair is a fairly simple process. However, these are detailed below, so that the student can be as safe and secure as possible while driving the platform.

Attaching the Wheelchair

The platform is meant to operate with the motors and wheels on the back of the platform, so that the user drives forward with nothing in front of them. To attach a wheelchair to the platform, the wheelchair must first be rolled up onto the platform. Make sure that the wheels align with the appropriate metal grooves, and that the wheelchair is as far back as it can comfortably go. Once this is complete, there are two metal handles with hooks attached to belts, that are found directly in front of the platform's wheels. Take out one of the handles, and attach it to an appropriate part of the opposite side of the wheelchair. Once attached, you can tighten the connection by pushing down on the metal plate on the back of the handle. Repeat this process with the other handle.

Detaching the Wheelchair

To detach a wheelchair from the platform, unhook the handles from where they are attached on the wheelchair. Next, push the metal plates on the sides of the handles' bases. This will cause the handles to retract back into the bases. After this is complete, the wheelchair can safely be rolled down and off of the platform.

Attaching the Platform to the Motors

To securely attach the wheelchair platform to the motors, first align the square metal prongs from the motors to the metal prongs of the platform. If done correctly, the ramp of the platform will be facing outward. Take out the metal fasteners, attached to the platform, by pulling them apart and then remove them completely from the square prongs. While the two different parts are still aligned, push them towards each other until there is a firm hold between them. Then reinstall the metal fasteners to the holes.

Controls

This product has the ability to be controlled using one input during a session as well as simultaneously using multiple inputs.

One Control Input
To make all other controls inoperable, you must use the interface to select which input you plan to use and press **SEND**. Remember to keep the computer attached during operation or else the other controls will still be fully functional.

Multiple Control Inputs
To accomplish multiple input control, do not connect the chair to the user interface. This will cancel any settings and allow you to use a few inputs to drive at the same time, but you will not be able to collect data.

Free Wheel Mode
Usually the platform should not be able to be moved by just pushing it. However, if you need to move the platform without driving or turning on the device, there is an option to do that. This is a mechanical feature of the chair and does not require access to the user interface.

Turn on Free Wheel Mode
Find the lever on the bottom of the device on the portion where the batteries are stored. The lever is specifically located close to the ground, in between the wheels, on the side where the power button is. Simultaneously pull on the lever and turn the handle clockwise until the lever is in a horizontal position.

Turn off Free Wheel Mode
Find the lever on the bottom of the device on the portion where the batteries are stored. The lever is specifically located close to the ground, in between the wheels, on the side where the power button is. Simultaneously pull on the lever and turn the handle counterclockwise until the lever is in a vertical position.

Maintenance
To properly operate the platform, first check to ensure that the emergency stop switch is not pressed. Furthermore, one needs to boot up the GUI and make sure the Arduino has been programmed with the correct code, so that the driving mode can be switched as needed. When transporting the platform from location to location, use free wheel mode as needed (see Configuration and Use for more details). Make sure that each component is securely connected to the platform and that there are no loose wires. To keep the platform up and running well, some maintenance can be done.

Table 1: Parts for Potential Replacement

Part Image and Name	Description	Link to Purchase
 **Battery**	The batteries are 12V sealed lead acid (SLA), with a rating of 55Ah. Any 12V SLA batteries with a rating of 45-55Ah will work with the platform. The current ones were purchased at Batteries Plus on Route 66 in Flagstaff, AZ.	Power Sonic 12V 55AH AGM SLA Battery with P Terminals - POWPS-12550P
 **Head Switch**	The wobble head switches are used as an alternative input for driving. The platform uses two of these for operation.	Store - AAC & Speech Devices from PRC
 **Thumb Joystick**	The thumb joystick is used as an alternative input for driving. The provided software allows for two different driving modes with the same joystick.	https://www.mouser.com/ProductDetail/SparkFun/COM-15168?qs=2WXlatMagcHW4UWWWoRwgw%3D%3D&gclid=EAIaIQobChMIjNjTzNui6QIVdx-tBh1tNgY-EAQYASABEgKzQ_D_BwE

Batteries

If used consistently, the batteries provided will last around 2-3 years (or 500-1000) charges. To disconnect the batteries, remove the bolt that holds the ring and wires in place, then slide the ring

off of the terminal. The platform requires two batteries to operate. Make sure to reconnect any wires disconnected through this process when installing new batteries.

Head Switches

If the head switches are no longer operational, they are simple to replace. Unplug the grey cable from the black adapter and replace with a new head switch. If they operate inconsistently, consult the Troubleshooting section before replacing them.

Joystick

The current platform uses a thumb joystick, which is soldered to the platform itself, and should not need to be replaced. If it does, one will need to desolder the jumper wires from the joystick board, bearing in mind the correct pins and connections (please consult the Arduino code for specific pins). When installing the new joystick, the wires will need to be resoldered to the new joystick board.

Arduino and GUI Code

The Arduino code used to run the microcontroller on the platform was written in version 1.8.5. The GUI code, developed in Processing, was written in version 3.5.3. Both programs should be compatible with future versions of the IDEs, but updates are not required. The codes are not meant to be changed, especially as they work with each other. With that in mind, if future teams would like to build upon the current implementation, it is recommended to download and read the programs from the team site before making any changes.

Troubleshooting Operation

Mobility

If the user is either stranded or stuck.

- An emergency release lever, located underneath the tail end of the rig, can be pulled and rotated 90 degrees clockwise to unlock the main wheels, allowing free movement, see free wheel mode (pg 10).

If the motors are completely unresponsive.

- Check if the combined battery voltage is less than 22V, if so, charge them.
- Check for error messages to see if Processing isn't recognizing the Arduino.
- Check if the Arduino is being powered by looking for illuminated LEDs, if not, the motor driver's 5V power supply needs to be reconnected to the arduino, refer to the schematic, soldering might be required.

If the motors are either sluggish or somewhat unresponsive.

- Batteries require charging, refer to the charging section under installation.

If either motor won't stop driving.

- A wire leading to the joystick has been knocked loose. If the wire has come loose at the base of the joystick, then it can be plugged back in by hand. Otherwise seek technical assistance as a wire will require the disassembly of most other electronics in order to be re-soldered.

If either head switch won't cause the rig to turn.

- In Processing, reset the control mode and re-select the headswitch option.
- Check if either wire is disconnected from the Arduino. If so, disassembly and soldering is required.

GUI

When all else fails, contact Dr. Winfree, Taylor, or another programmer to look into the issue. But make sure that you have exhausted all the options in this section first.

If the interface window doesn't pop up/freezes when I run the code.

- Most likely, an error in the code has occurred that prevents it from running. Check the console window when trying to run the code. If there are errors, they will appear in red text on the console, and can be used to debug the code.

    - If you are not confident in your programming ability, make sure that you downloaded the files directly from the team site and did not change them.
- Make sure that the serial monitor in the Arduino IDE is not open.
- Unplug the Arduino from the computer's USB and try a different port.

If the data that was supposed to be saved cannot be found.

- Make sure that you followed the directions in the Configuration and Use section.
- Check the "AmpEd_GUI" folder where the Processing code is stored. This is where the data file will land once it has been created.
- Make sure that data has been displayed in the "RESULTS" box on the right side of the screen.
    - This indicates a successful save.
    - The file is not made until the **STOP RECORDING** button has been clicked and the results have been displayed.

If the chosen driving mode selection doesn't work.

- Download the test code from the team site, and follow steps 6 and 9 to upload the code to the Arduino. Once this has been done, run the interface code. You can skip the student and tech name, but make your selection and hit **SEND**. Once this has been done, check the yellow LED on the Arduino. You should see a certain behavior, depending on what you chose.
    - Regular joystick: onboard LED blinks at 1 second intervals
    - Unidirectional joystick: onboard LED blinks at 2 second intervals
    - Head switches: onboard LED blinks at 3 second intervals
    - If none of these behaviors work, hit the red reset button on the Arduino itself, and then re-upload the Arduino test code

Conclusion

Our team is proud to bring you our user friendly and powerful augmented mobility platform. We hope that you are satisfied with the product and able to use it for many years to come. We are glad that we could assist you in your goal to help children with mobility issues to experience independent mobility, and hope that the students are, too. Although the school year will be ending soon, we want to make sure your experience with our product is smooth, especially in the first few months of operation. Our NAU email addresses are included on the cover page of this manual, should you need to contact us with any questions, comments, or concerns. Please do not hesitate to reach out to us or Dr. Winfree if you need anything.

With best wishes from your product developers,

Appendix A: System Flowchart

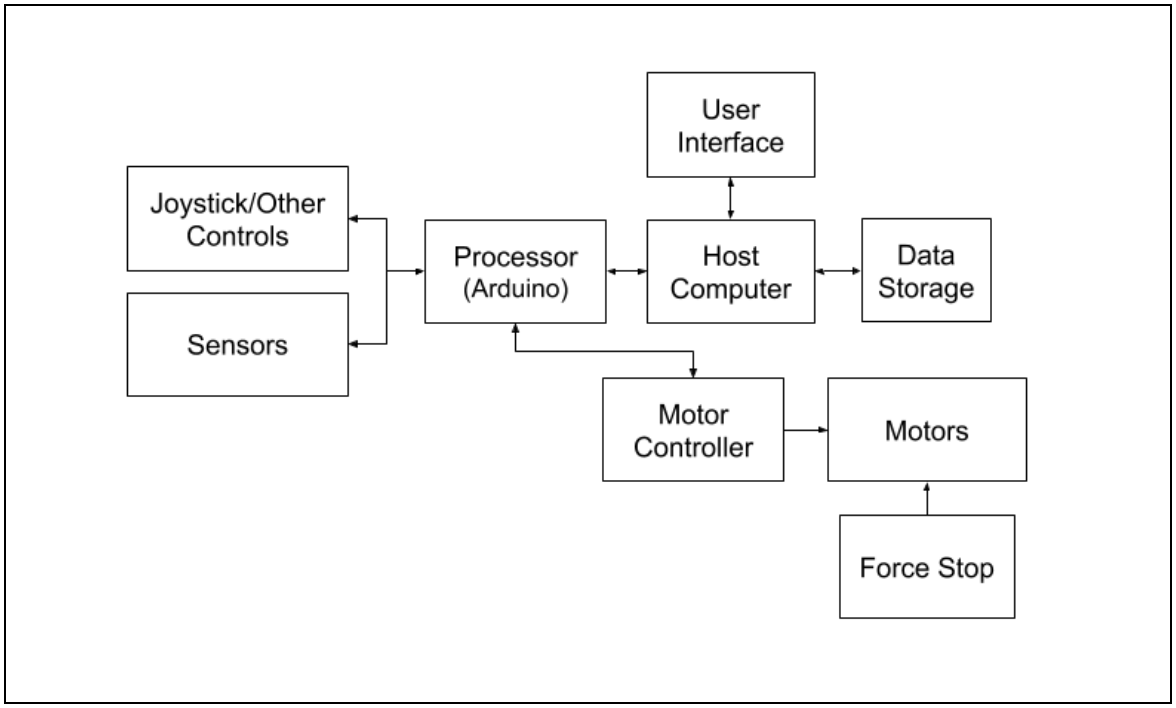

Figure 12: System and data flowchart

Appendix B: System Schematic

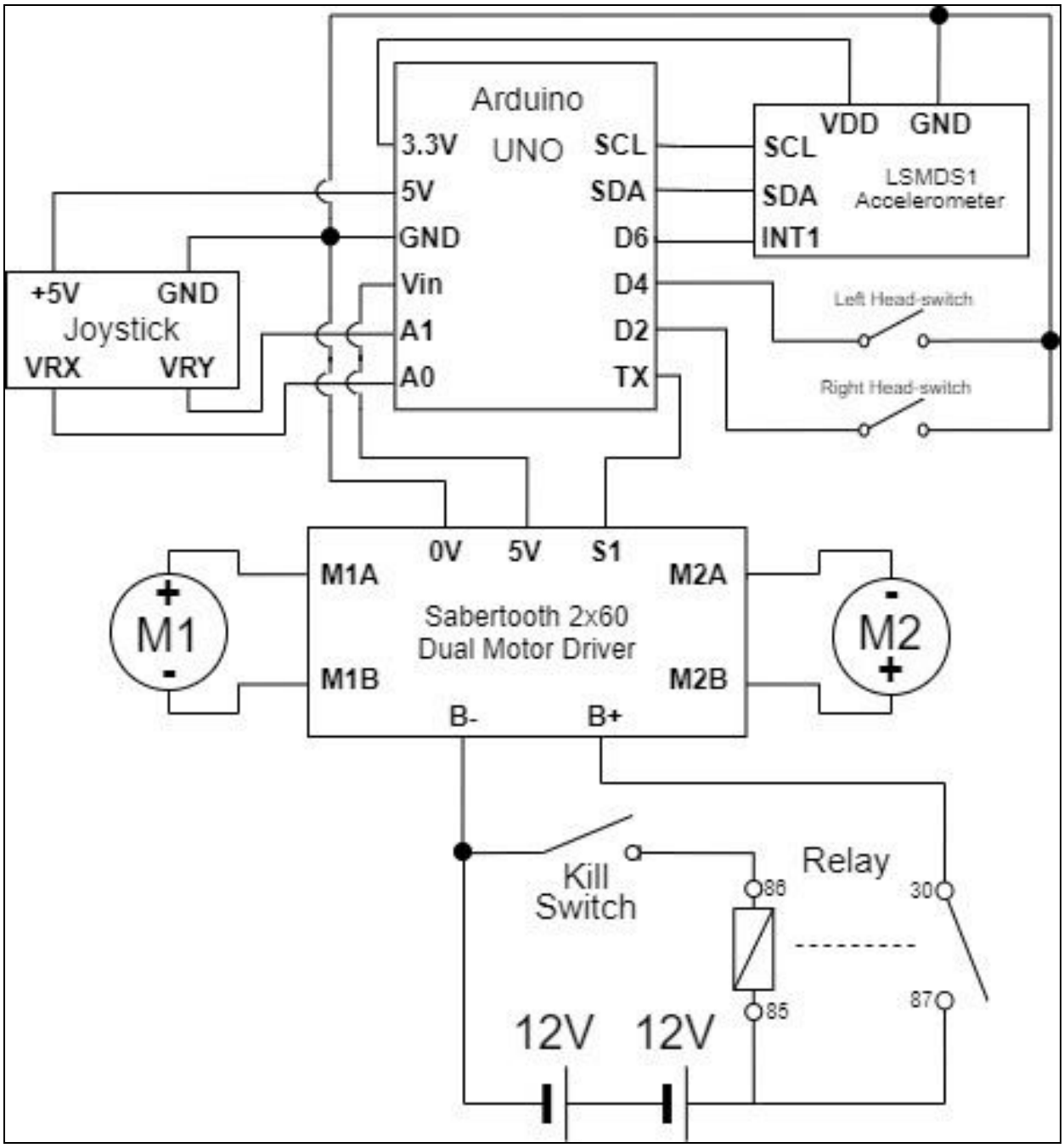

Figure 13: Onboard circuitry schematic