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

Project:
Augmented Powered Mobility

DR3 Group Report

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

The 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 mechanical engineering students at NAU to develop the Augmented Powered Mobility (APM) platform, but currently needs a more robust, adaptable, and updated platform in order to meet the needs of her patients.

The Project

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 force feedback training. 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. 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.

WBS Discussion

At the outset of the project, the team determined three primary subsystems, or areas of focus, that would be crucial to the success of the project. Per the structure of a work breakdown structure (WBS), these subsystems were then broken down into components, in order to better visualize what needed to be done so that each subsystem could be functional by early April. These can be seen in the team's WBS charts in Figures 1-3.

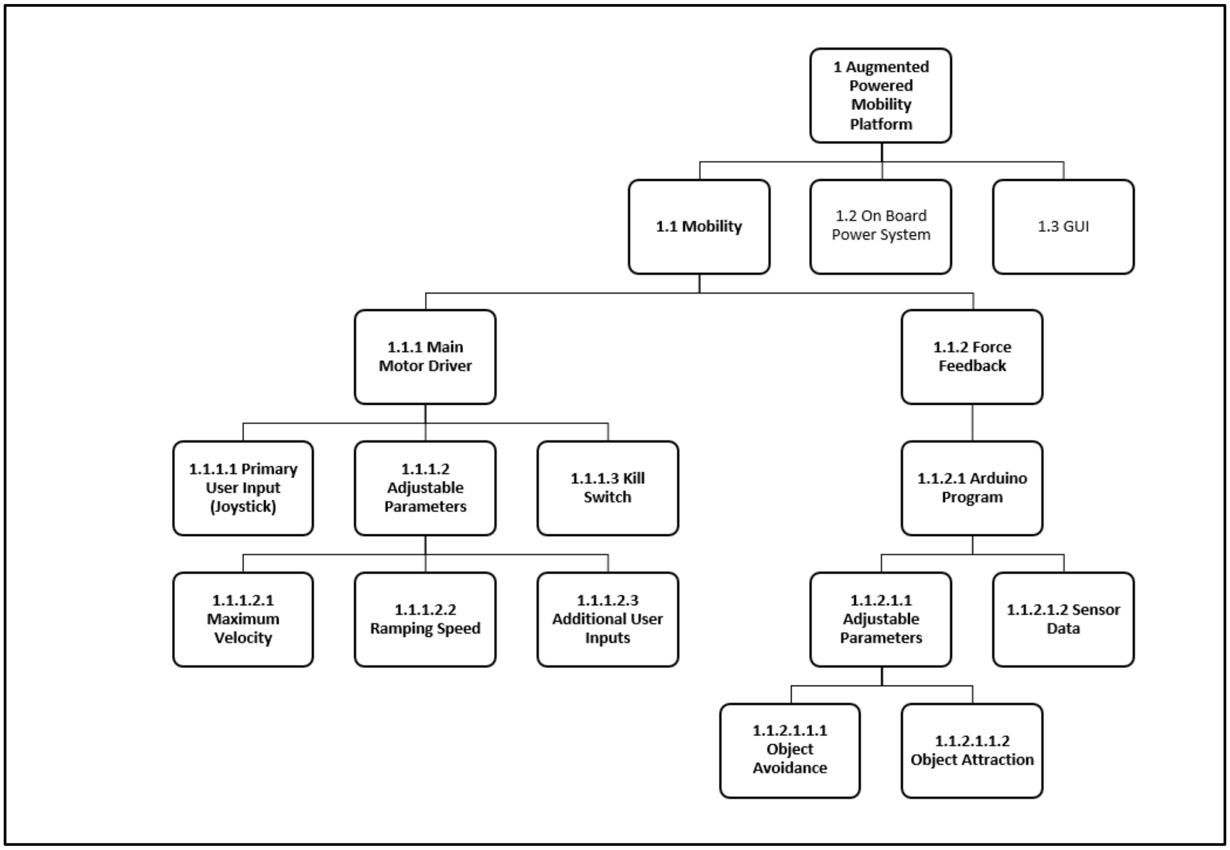

Figure 1: Detailed mobility subsystem

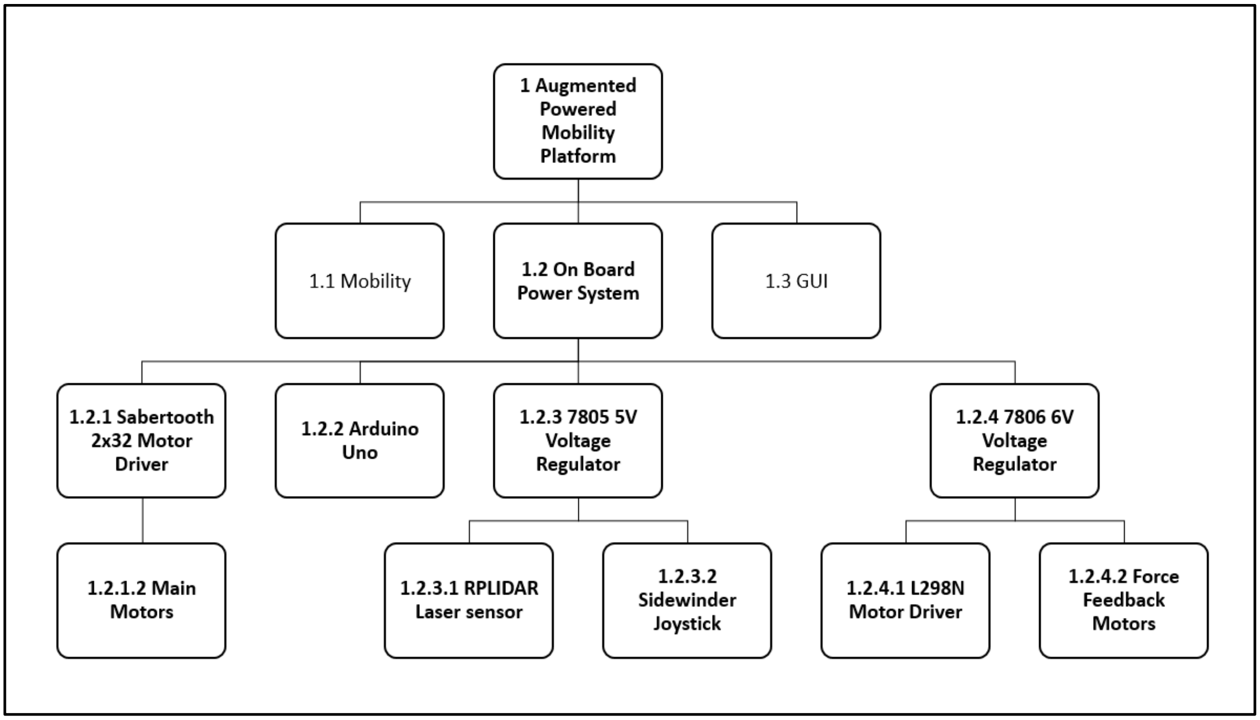

Figure 2: Detailed power subsystem

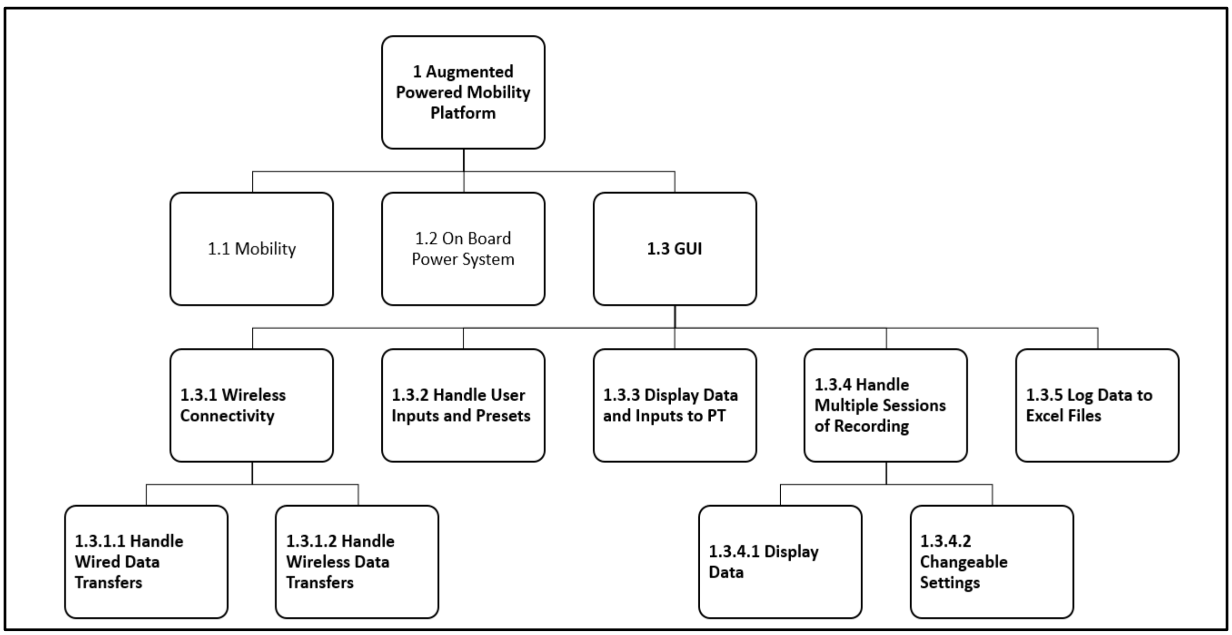

Figure 3: Detailed GUI subsystem

Mobility Subsystem

Denoted as 1.1, the mobility subsystem focuses on making sure the platform itself can “talk” to, and move in conjunction with, the patient’s EPW. This includes implementing a force feedback joystick, as well as other adjustable parameters and switches, that can be used by a variety of patents. This subsystem will primarily be run by an Arduino Uno, which can send control signals to each component, and handle resulting data outputs accordingly.

On-Board Power Subsystem

Denoted as 1.2, the on-board power subsystem is devoted to ensuring that the team's new platform can run off of two 12V batteries. Important parts include the motors, Arduinos, 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.

GUI Subsystem

Denoted as 1.3, the GUI subsystem, which this status report will primarily focus on, is responsible for displaying requested real-time data to the physical therapist in a way that is easy to read (hence the term "graphical"). Furthermore, the GUI handles data logging and user presets that the PT would like to be able to change and reset for each patient that she works with. This will be done primarily in the Processing IDE and requires the most time and resources.

Mobility Subsystem Updates

As determined by the team, the mobility subsystem consists of two major components, or outcomes, that are crucial to its success. First, it should implement the main motor driver, so that the platform itself can move in response to the user's joystick input. It should also have a "kill switch", or an emergency stop button to power the system down if needed. Second, it should implement a force feedback joystick, with possible mode selection between object avoidance and object attraction.

The original Gantt chart had the mobility subsystem slated to be done by mid-March, so that we could field test leading up to the UGrads presentation in April. However, this was unrealistic, so we have instead pushed the mobility subsystem completion date back to early April. Risk for delayed completion of the mobility subsystem is fairly high, as our platform needs to be able to move in order to perform the real-time data collection that our client wants to view later. Progress in regard to each component is discussed below, including their accompanying challenges, contingencies, and resolutions.

1.1.1 Main Motor Driver

As projected by the mobility Gantt chart in our DR2, the entire rig should have been mobile by January 26th. This is not the case as several setbacks, paired with weeks' worth of shipping time, has delayed progress. Firstly, the motor driver needed to be upgraded as it was determined that each of the two motors were pulling the peak possible current, 60A. Along with switching to the Sabertooth 2x60 motor driver, 6 AWG wire and a 200A 12V automotive relay were ordered to accommodate for the high wattage. Another unforeseen challenge was how different the new motor driver operated in regard to its previous model. Using different methods to code movement, as well as the company's software, DEscribe, lacking crucial functions for changing how the motor driver operates in comparison to its previous models. New code has been written for the mixed differential drive using a downloaded Sabertooth Arduino library, though it can't be tested until the rig is sustainably mobile. Before that can happen, another relay needs to be ordered, since the rig cannot be powered by 12V. As well as the wires leading from the motor should be upgraded to 6 AWG to prevent overheating.

1.1.2 Force Feedback

Regarding the force feedback Gantt chart in our DR2, we are ahead of schedule. Power is stepped down from the 24V battery to 5V for each of the two internal motors, as well as for the joystick analog values. Code has been written to have the controller stabilize to the center, to prevent accidental user input. While the rig is not mobile, the force feedback controller has been tested using three unmounted infrared proximity sensors with code that will repel the joystick away from the direction of any detected object in those directions. The challenge with force feedback is finding an appropriate scale for how much force to apply to the controller depending on distance to objects.

Power Subsystem Updates

As determined by the team, the power subsystem consists of four major components, or outcomes, that are crucial to its success. First, it should implement the Sabertooth motor driver in order to power the main motors. Second, it should be able to properly step down the voltage drawn from the two 12V lead acid batteries provided, in order to power the Arduino Uno necessary for data collection. Third, it should implement the 7805 5V voltage regulator, in order to provide power to the RPLIDAR laser sensors and Sidewinder force feedback joystick. Fourth, it should implement the 7806 6V voltage regulator if time allows. The 7806 performs the same tasks as the 7805, but steps the voltage down to 6V instead of 5V.

The original Gantt chart had the power subsystem slated to be done by mid-March, so that we could field test leading up to the UGrads presentation in April. This subsystem is actually ahead of schedule, meaning it is essentially completed. Risk for delayed completion of the power subsystem is fairly high, as our platform needs to be powered properly in order to run diagnostics checks, and provide sufficient power to the mobility and GUI subsystems. Progress in regard to the components collectively is discussed below, including their accompanying challenges, contingencies, and resolutions.

Regarding the Power Gantt chart in our DR2, we are ahead of schedule. That means that a circuit has been designed and tested that powers several proximity sensors, an Arduino Uno, and the Sabertooth 2x60 motor driver simultaneously. A circuit has been designed for charging the batteries, but this was created with parallel batteries in mind. Since the battery charger is only rated for 12V, in order to charge the two 12V batteries in series, a 24V boost converter is going to be needed to achieve charging, though this will negatively affect charging time.

GUI Subsystem Updates

As determined by the team, the GUI subsystem consists of five major components, or outcomes, that are crucial to its success. First, it should be able to handle wireless connectivity, to minimize the chances of faulty and incorrectly placed or routed wires. Second, it should be able to take in input from the PT and adjust parameters accordingly. Third, it should display the desired inputs that the PT selected, and the resulting data and metrics gathered from the patient's performance. Fourth, it should be able to handle multiple sessions of recording and have a way of differentiating recordings for different patients. Fifth, it should have a way of logging specific data to a file in a manner suitable for the client to read.

The original Gantt chart had the GUI slated to be done by mid-March, so that we could field test leading up to the UGrads presentation in April. However, this was unrealistic, so we have instead pushed the GUI subsystem completion date back to early April. Risk for delayed completion of the GUI, however, is lower than the other subsystems, as the platform itself can function without a fully working GUI. Progress in regard to each component is discussed below, including their accompanying challenges, contingencies, and resolutions.

1.3.1 Wireless Connectivity

As mentioned before, the subsystem should be able to run at least somewhat wirelessly. This does not mean completely eliminating the use of wires, but rather, aims to minimize their usage as much as is realistically possible. The team decided to pursue Bluetooth connections, as opposed to WiFi. This component, however, was determined to be a lower priority in comparison to components 1.3.2, 1.3.3, and 1.3.5. As a result, different Bluetooth transceivers have been researched and discussed for implementation in the project, but nothing concrete has been done here. However, as other components and the other subsystems draw nearer to completion, more time and resources can be re-allocated here. This component is still on schedule, as we did not plan to have this done until mid-March.

1.3.2 Handle User Inputs and Presets

The GUI should be able to handle user inputs and presets. For example, if the client would like to set a maximum motor speed for the patient's safety, or increase the strength of the force feedback joystick, the GUI should handle this and send control signals to the Arduino to set. This component works in conjunction with 1.3.3. As of now, some preliminary sketches have been built through Processing that can handle mouse clicks and single-key inputs. Next steps to progress with this component will include building a more robust system module that can handle a string of keystrokes or map the single-key inputs to more complex functions. We are behind on this component, as we had hoped to have this working in early January. However, now that we have more information and research completed, progress here should be exponential.

1.3.3 Display Data and Inputs to PT

At minimum, the completed subsystem should display the inputs that the user has selected, as well as real-time, constantly refreshing, data outputs. For example, if the client would like to see that a recording session is occurring, the interface should reflect that. If she would like to view the current speed of the EPW, or the number of times the patient has collided with an object since the session began, these metrics should be viewable from her laptop or tablet screen. This component is reliant on the functionality of 1.3.2, and as such, not much progress here has been made, aside from what has been done in 1.3.2. However, code has been produced that can show real-time data from a single sensor, so work has been done here. Next steps include plotting real-time data from multiple sensors simultaneously, as well as real-time accompanying calculations as needed. Although we thought that displaying the data would be done by late January, we have encountered some difficulties with getting the proper parts needed for the platform. As such, progress here has been pushed back. However, we have ordered and received some of these, so we are confident that we can make up the lost time.

1.3.4 Handle Multiple Sessions of Recording

The GUI should be able to handle multiple sessions of recording. For example, if the client were to work with multiple patients, she should be able to choose who she is recording a new session for. This should be handled internally by the GUI, but also displayed externally so that she can tell which “profile” or “patient” she is accessing. As of now, the client only has one patient in mind, but the goal is to create an adaptable platform that can be used in the future. This component depends heavily on the functionality of 1.3.2 and 1.3.3, and as such, no substantial work has been done on this component. However, some work done on 1.3.5 will translate over to this component. Next steps include creating an option to allow for separate recording sessions, as well as mapping user inputs to this selection process. We had planned to have this component done in early February, so we are a little bit behind. However, since we can devote more attention to this aspect of the project, we are confident that we will complete it by early to mid-March.

1.3.5 Log Data to Excel Files

Finally, the completed subsystem should be able to log sensor data, important readings, and the date and time of each set in an appropriately named file and filetype. This file should also read in such a way that someone who is not an engineer would understand what the data means. For example, if one column reads “27 13 9 7 2”, the file should have identifiers so that the user knows that these numbers represent distances from an object, and could possibly be able to graph data automatically, so that the user does not have to. This component has the most progress, as wired circuits and code have been produced that show proper data logging with the correct data and time stamps. The data has been stored in a comma-separated values (.csv) file on a microSD card, which can be read on a computer through Microsoft Excel. Next steps regarding this component include testing it with a functional displaying GUI, as well as running it alongside the full platform, as opposed to a simple sensor network. This component is well ahead of schedule, as we were not originally planning on finishing work here until mid-March.

Conclusion

In the mobility subsystem, there are three major components: the motor driver, force feedback and adaptability. Currently the motor driver relies on user input to operate the chair with a hardwired killswitch that cuts power to all systems. Varying how the analog user input is mapped to PWM values being sent to the motor driver, allows for fine tuning the acceleration in order to improve user comfort. For adaptability, the chair will have multiple different control schemes in order to support a wider range of physical disabilities. The original completion timeframe was mid-March, however, after several unforeseen delays, dead ends, and meeting with the client, we adjusted the vision for the mobility subsystem and extended our due date to early April.

The completion of the power subsystem is crucial to realizing self-sufficiency. All components must be powered from the onboard 12V batteries including any sensors, microprocessors and motors. The subsystem must be able to appropriately step down the voltage to power the low power components such as the Arduino and all sensors. When the batteries are low, there should be a simple way to recharge and all power to the other subsystems needs to halt while charging. The original anticipated completion date was mid-March, but was slightly changed due to the environment surrounding COVID-19, however, major progress has been made and should not be delayed further.

The GUI presents challenges of its own. Although there is not much to physically show, a great deal of background research has been done, and preliminary programs have been written. These programs have been written at varying levels of complexity, but ultimately, these programs need to be expanded and modified so that they can run with the team's full platform. Since much of the preliminary research has been done, the primary goal of the second half of the semester is to get code up and running and integrate parts. The biggest challenge that the team will face with the GUI will be to integrate the code written for a small sensor network, with a full-size platform. This will include debugging, testing, and refactoring. However, with the progress made to this point, we as a team are confident that the project will be done on time.