December 2021

Soft Continuum Robotic Airbag Integrated with Passive Walker for Fall Mitigation

Jacob Andrew Thompson
Clemson University, jacobthompson127@gmail.com

Soft Continuum Robotic Airbag Integrated with Passive Walker for Fall Mitigation

---

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Electrical Engineering

---

by
Jacob Thompson
December 2021

Accepted by:
Dr. Ian Walker, Committee Chair
Dr. Ge Lv
Dr. Richard Groff

Abstract

This thesis describes the prototype and development of a soft continuum robotic airbag system that is attached to a passive mobility walker. The system can deploy in multiple configurations: to the front, left, or right of the walker depending on the direction of a detected fall. The continuum component of the project is made of nylon fabric with thin cables, allowing it to be compactly stored before deploying. The airbag is inflated in real time during a fall using a novel compression system. Results of experiments with the prototype in each configuration are presented. The system deploys consistently across falls, significantly reducing the g-force of impact.

Acknowledgments

Firstly, I would like to thank Dr. Ian Walker for his support and guidance throughout this project. His creativity and open-mindedness helped move this project forward and inspired me to think outside the box on a multitude of other projects.

I would also like to thank Dr. Ge Lv and Dr. Richard Groff for taking the time out of their busy schedules to serve on my defense committee. Additionally, I would like to thank all the professors that I have had the pleasure of learning from as well as Clemson University for the many opportunities it provides. I also want to thank both past and present members of our research group, who have all been eager to share their knowledge with me.

Lastly, I want to thank my friends and family for their support throughout my educational career. Their encouragement has been inspirational in keeping me motivated and determined to further my academic goals. The tolerance of my roommates for this project's footprint and clutter was generous and greatly appreciated.

Table of Contents

Title Page . . . . . **i**

Abstract . . . . . **ii**

Acknowledgments . . . . . **iii**

List of Tables . . . . . **v**

List of Figures . . . . . **vi**

1 Introduction **1**
1.1 Overview 1
1.2 Background and Related Work 1
1.3 Thesis Overview 5

2 System Design and Prototyping **6**
2.1 Preliminary Inflation Research and Development 6
2.2 Arm and Bladder Construction 10
2.3 Mechanical and Hardware 16
2.4 Integration with Walker and Actuators 18
2.5 Electrical Design and Build 26
2.6 Software 37

3 Results Using the Prototype **41**
3.1 Preliminary Deploy Tests 41
3.2 Final Deploy Tests 55
3.3 Variable Weight Tests 62

4 Conclusions and Discussion **65**
4.1 Future Work 66

Appendices **67**
A Alternative Inflation Methods 68
B Software Details 74

Bibliography **86**

List of Tables

2.1 Major electronic components 26
2.2 Encoder values to engage brakes for different arm configurations 39

3.1 Summary of experimental results - peak g-force measured during each fall 62
3.2 Summary of variable weight results - peak g-force measured during the initial impact with the ground for each fall 64

List of Figures

1.1 Early smart walker systems with disc brakes (left) and center of gravity positioning (right) . . . 2
1.2 JAIST Active Robotic Walker (left) and ISR-AIWALKER (right) that each help with maneuverability and obstacle avoidance . . . 2
1.3 Hit-Air inflatable pull-cord vest (left) and Helite wearable airbag for cycling (right) . 3
1.4 Human airbag systems deploying . . . 4
1.5 Several pneumatic continuum robots . . . 5

2.1 Prototype using bungees to squish bladder between two rigid plates . . . 7
2.2 Prototype with a square rod spun by a drill . . . 8
2.3 Deflection in prototype with 3/8″ rod . . . 9
2.4 Prototype with 3/4″ rod and spacers . . . 9
2.5 Small-scale prototype deploy test . . . 10
2.6 Adhesive side of the fabric (left) and uncoated side (right) . . . 11
2.7 Identical fabric pieces cut out for the arm and bladder . . . 11
2.8 Air nozzle sealed to the arm . . . 12
2.9 First full inflation of the fabric arm and air bladder . . . 14
2.10 Components of the end of the arm . . . 15
2.11 Relative dimensions and positions of fabric arm components . . . 16
2.12 Front view of the whole system . . . 17
2.13 Cutting the slots for straps (left) and finished front plate on the walker (right) . . . 18
2.14 Smooth guides for the webbing straps on the 3D printer (left) and partially installed on the front plate (right) . . . 19
2.15 Adapter with embedded hex bit for connecting drill and square rod . . . 19
2.16 Spinning shaft components labeled . . . 20
2.17 Overview of walker rear side . . . 22
2.18 The completed motor assembly for one arm cable . . . 23
2.19 All motor assemblies mounted on the walker . . . 24
2.20 Main electrical control systems for the project on the right side of the walker frame . 27
2.21 Voltage regulation circuit diagram . . . 28
2.22 Overall microcontroller pinout diagram . . . 29
2.23 Electrical control panel . . . 30
2.24 The wiring diagram of a power drill trigger. A SPDT switch selects the forward or reverse direction, a potentiometer controls the speed, and a SPST switch signals if the trigger is pulled or not. . . . 31
2.25 Drill trigger replacement circuit diagram . . . 32
2.26 X9C102 digital potentiometer block diagram . . . 32
2.27 GPIO expander circuit diagram . . . 33
2.28 Electromagnetic motor brakes circuit diagram . . . 34
2.29 Inside the electrical control box with relays and digital potentiometers to interface with the brakes and drill. . . . 34

2.30 DRV8825 motor driver pinout for our system 35
2.31 Encoder and level shifter circuit diagram 35
2.32 Cable assemblies as referenced in the software 38

3.1 Deploy test with speed set to 2 and clutch on maximum. There is still air in the bladder that should be forced into the arm, but the drill stopped spinning before the arm was sufficiently pressurized. 42
3.2 Force diagram of the drill when it spins clockwise (top) vs counterclockwise (bottom). Later tests were run with the drill spinning counterclockwise so that the force pushed the handle into the wood instead of pulling on the thin plastic zip ties. 43
3.3 Deploying by driving the motors to position (left) vs letting the arm pull them to position (right). This shows the motors can move faster than the arm pulls them out, so either deploy method is an option. 44
3.4 Motor cables tangled and wrapped up after driving the motors to position at max speed 45
3.5 Encoder positions using hybrid deploy method 46
3.6 Improved motor assembly with dual-shaft stepper, electromagnetic brake, and stronger mount 46
3.7 Encoder values after upgraded motor assembly 47
3.8 Spiral cable wrap spacers 48
3.9 Selection of different configurations that the arm with spacers can achieve 49
3.10 Data collection modes in Physics Toolbox app 50
3.11 Phone mount locations for impact data collection 51
3.12 G-Force data from baseline tests of just the dummy with ankle weights falling onto the futon cushion 52
3.13 G-Force data from baseline tests with the walker and the dummy, showing worse results than without the walker 53
3.14 Separation at time of impact causing worse impact with the walker than without the walker 54
3.15 Data logging - g-forces experienced by the dummy during baseline left tests 56
3.16 Progression of typical baseline left fall 57
3.17 Baseline forward tests g-forces 57
3.18 Progression of typical baseline forward fall 58
3.19 Left deploy g-forces over five tests 59
3.20 Progression of a left fall with the system deploying. 59
3.21 Right deploy g-forces over five tests 60
3.22 Progression of a right fall with the system deploying. 60
3.23 Forward deploy g-forces over five tests 61
3.24 Progression of a forward fall with the system deploying. 61
3.25 Weighted tests with 25 pounds added to the dummy 63
3.26 Weighted tests with 50 pounds added to the dummy 63
3.27 Progression of a forward fall with additional weight in the backpacks 64

A.1 Size comparison of compressed gas cartridges 70

Chapter 1

Introduction

1.1 Overview

Falls are the second leading cause of injury deaths worldwide [1], and the leading cause of injury and death in older adults in the United States [2]. In 2018, 27.5% of U.S. adults aged 65 years or over reported at least one fall in the past year (35.6 million falls). Over 10% (8.4 million) reported a fall-related injury, resulting in an estimated 3 million emergency department visits, more than 950,000 hospitalizations, and approximately 32,000 deaths [3]. More than 95% of hip fractures are caused by falls [2]. Fear of falls has been found to be a significant risk factor limiting activity in the elderly [4].

Falls are more likely, and often more critical, for those individuals who are fragile and use walkers. In a recent study, it was found that people using a walker were 7 times more likely to be injured by a fall compared to those using a cane [5]. Passive walkers help provide stability for such individuals during ambulation, but offer no protection in the event of a fall. This thesis discusses the augmentation of a passive walker with a robotic airbag system to mitigate the effect of falls.

1.2 Background and Related Work

The concept of creating robotic versions of passive walkers, or "smart walkers," is not new. An early smart walker was a modified rollator walker developed by the Dublin Institute of Technology using hydraulic disc brakes [6]. This system used a high-speed linear actuator to vary the pressure

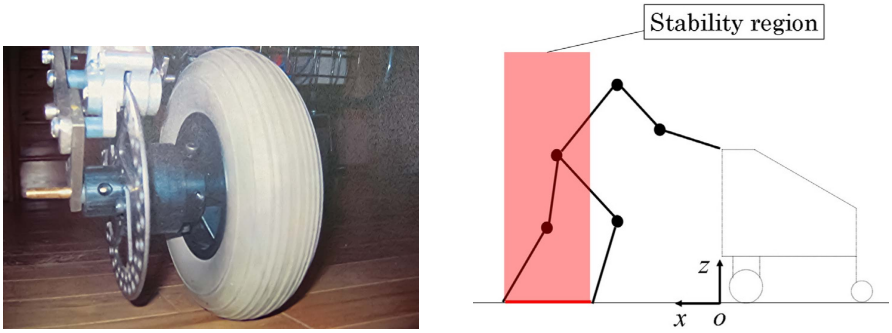


Figure 1.1: Early smart walker systems with disc brakes (left) and center of gravity positioning (right)

on the braking disk. The brake activated if the user moved too fast or if the walker sensed the user wanted to stop. Another rollator walker with pneumatic brakes on the wheels [7] simulated three different falling situations - freezing of limbs, stumble, and loss of balance. Another walker that implements fall prevention is the RT Walker [8], which focuses on calculating the user's center of gravity in real time. When the user's center of gravity is detected to be outside the region of stability, brakes are enabled. These systems are illustrated in Figure 1.1. One drawback of these braking systems is that they do not address cases when the walker is tilting or falling to the left or the right.

Some smart walkers aim to help the user in a more subtle way. A robotic walker developed by the Japan Advanced Institute of Science and Technology (JAIST) autonomously adjusts its direction and speed according to the user's walking movements [9]. It does not require any user input, instead using a pair of laser range finders to detect the user's legs and any obstacles in the environment.

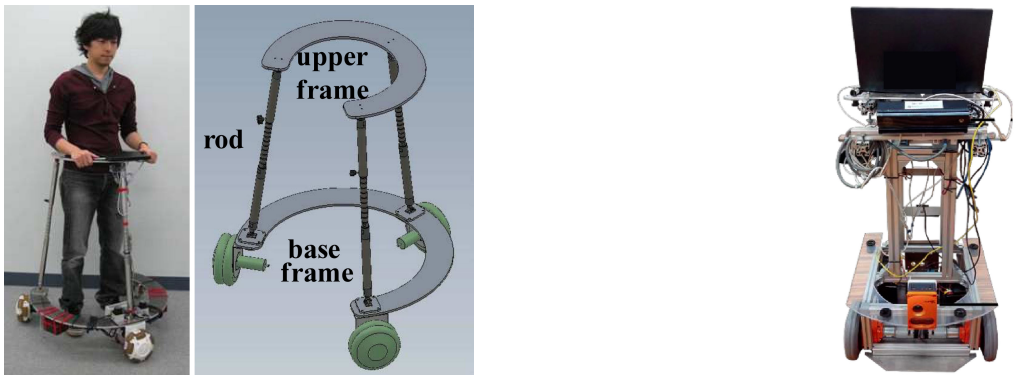


Figure 1.2: JAIST Active Robotic Walker (left) and ISR-AIWALKER (right) that each help with maneuverability and obstacle avoidance

The walker uses a trio of Mecanum wheels to drive the walker away from obstructions in an indoor environment, while still following the general direction the user wants to go. This robotic walker provides a measure of fall prevention by avoiding obstacles or drop-offs that could unbalance the user, but in the case of a fall it simply locks up the wheels.

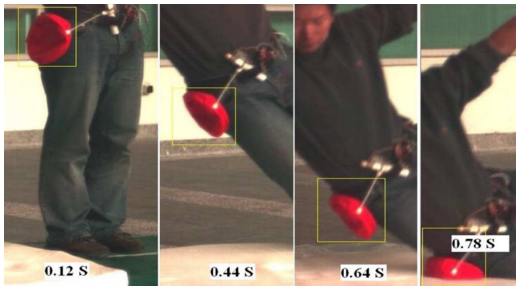
A similar robotic walker platform is the ISR-AIWALKER. A model built to loosely resemble a standard walker, it uses two motorized wheels and two castor wheels to drive around. The system infers user intention through a pair of force-sensitive handles, Leap Motion sensors, and several other sensors for gait analysis. All of these inputs are aggregated and used to maneuver the walker in tandem with the user's intents. These smart walker systems can be seen in Figure 1.2.

Several groups have worked on "human airbags" - wearable devices which deploy when the wearer is falling. These devices come in many forms such as vests, belts [10], jackets, or harnesses [11]. Helite is a company that has made wearable airbag vests for skiing [12] and cycling [13]. The Hit-Air inflatable air vest [14] is a body-worn airbag with a pull cord marketed towards equestrians, where the rip cord is attached to the horse's saddle and is pulled if they fall off the horse. See Figure 1.3. These commercially available products demonstrate a need for personal fall protection in a variety of environments.

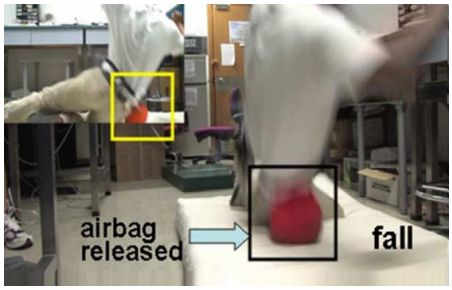
Figure 1.3: Hit-Air inflatable pull-cord vest (left) and Helite wearable airbag for cycling (right)

Many human airbag systems for fall protection are designed as a belt or fanny pack that the user must wear all the time. For example, the system discussed in [15] involves a belt with a microcontroller, inertial measurement unit (IMU), compressed CO_2 cylinder, and actuation mechanism with airbags around the user's hip. When the IMU detects the person is falling, it punctures the gas cylinder and rapidly inflates the small airbag in front of the user's hip within 333ms. A demonstration of this device is shown in Figure 1.4a. A similar human airbag worn as a belt ws

designed in [16], and built in [17] with a focus on microelectromechanical systems (MEMS) to detect the fall. The MEMS detection based system is shown deploying in Figure 1.4b.



(a) Small hip airbag deploying



(b) MEMS based human airbag deploying

Figure 1.4: Human airbag systems deploying

However, these systems are static, in the sense that the deployed system remains the same regardless of the specific fall endured by the individual. In this thesis, we introduce a novel alternative: a reconfigurable robotic airbag system featuring the ability to deploy the airbag system in the direction most needed. This provides the unique ability to tailor the response to the situation encountered, e.g. maximizing the volume of airbag between the falling individual and the environment.

The key objective in this work was to develop a novel airbag deployment system, integrated with a walker, which is not static but instead robotic, i.e. with programmable configurability, varying to meet the deployment needs of a specific fall. The goal was for the airbag to be stowed in one face of the walker, but be deployable in real time, as a function of the direction of a detected fall, across the range of a semi-cylindrical region from the left side of, around the front of, and to the right side of, the walker.

In order to address the above specifications, the core element of the design selected was an inflatable continuum arm, actuated (configured) in real time by remotely actuated tendons. Continuum (continuous backbone) robots have been the subject of much interest and research in the past few years [18]. Continuum robots have been successfully applied to a variety of medical procedures [19]. Inspired in large part by emerging research in soft robotics [20], [21], [22], researchers have been exploring the shaping of soft air-filled volumes using tendons to create continuum robots [23].

Numerous tendon-actuated pneumatic continuum robots have been demonstrated [24], [25].

Kinetic Sciences Inc. (KSI) developed a robot (Figure 1.5a) powered by pneumatic bellows and electric motors with cable guides [26]. Researchers at Clemson developed the Air-Octor (Figure 1.5b), which used pneumatically pressurized central chambers to provide structure for the arm while tendons attached to an exterior section controlled the curvature [27]. One pneumatic robot (Figure 1.5c) navigates its environment through pressure-driven eversion [28]. One chamber is pressurized more than the other chamber, and the robot bends away from the higher pressure chamber. The design described in this paper expands on our experience in the area of continuum robotics [27]. It is the first application of continuum robotics to airbag technology.

(a) KSI Tentacle Robot (b) Air-Octor (c) Navigate through Growth

Figure 1.5: Several pneumatic continuum robots

1.3 Thesis Overview

The chapters that follow describe the journey of designing, building, improving, and testing the soft airbag system on a walker frame. Chapter two describes the construction of the fabric part of the system, along with early prototype tests for the actuation method. The final hardware and software of the finished system are presented. Chapter three discusses improvements on the design and data collection methods of the system, and presents the results of a set of experiments. Chapter four contains conclusions reached about the project as a whole and suggestions for future work.

Chapter 2

System Design and Prototyping

In this chapter, we discuss the iterative development process of building our final system. First, we cover early prototypes, the construction of the fabric arm, and finally the hardware and software in the final system.

2.1 Preliminary Inflation Research and Development

Before building any prototypes, a variety of air deployment methods were investigated. We considered using CO_2, high pressure air (HPA), and vehicle airbags. Detailed calculations and discussions for these alternate inflation methods is in Appendix A. Each of these methods have their own merits, but each had significant drawbacks for the prototyping of a continuum system. Using CO_2 canisters would have cost around $8 per deploy, and it would have been challenging to make a system that perfectly inflated upon cartridge puncture but that did not over-inflate (and explode) or under-inflate (and not effectively catch the user). Similar issues exist with HPA, along with concerns of the time required to release a sufficient volume of HPA into the arm. Airbag cartridges are very dangerous, produce toxic chemicals, and expensive ($200+ per deploy).

Consequently, we selected a method of inflation based on volume displacement. We planned to construct one section that would hold the air before the system deployed, and have that section be continuous with the arm section of the system. The arm would deploy when air was pushed from one place to another inside the sealed container.

2.1.1 Early deployment methods considered

The first prototype of the system with the above concept used bungee cords to pull two plates together with an air bladder between them as shown in Figure 2.1. It used a 6.0″ (15.2 cm) tall tower at each corner of the squish plates to hold the plates apart. Each tower had a servo motor attached to a metal rod which extended under the plate. The servo motors simultaneously pulled the rods out from under the plates and the bungees rapidly pulled the two plates together, forcing the air from between the plates to the section of the air container outside of the plates. This system deployed very quickly and was able to initially inflate the volume. However, the bungees applied the most force when they were under the most tension at the very beginning of deployment. Our application needs the most force at the end of the deployment when the user is falling on the arm or the arm is pushing off an object to stabilize the walker. Additionally, this prototype was very heavy (more than 20 lbs (9 kg)) because its frame was made of 4040 aluminum extrusion. Future iterations of this might have used square aluminum tubing or 90° aluminum pieces for the frame of the rigid plates and an inelastic material in the middle, but would still ultimately be fairly heavy because of the stiffness requirements for the four towers and the constraint that the two plates have to be larger than the air bladder. These drawbacks led us to investigate other deployment methods.

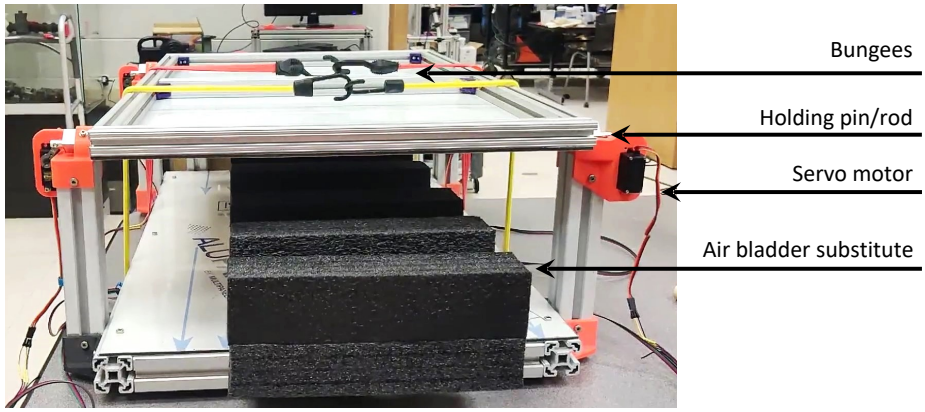


Figure 2.1: Prototype using bungees to squish bladder between two rigid plates

One option considered was to attach the inflated air bladder to a shaft and roll up the bladder itself. An advantage of this method is simplicity. However, we conjectured that attaching the bladder to a shaft would result in a very uneven force distribution. Further, attaching the inelastic fabric to the strap seemed likely to damage the fabric when spinning.

After the non-success of the bungee method, we investigated other methods to rapidly

compress a flexible container. We came up with the concept of putting a "cage" of straps around the air bladder and tightening the cage to compress it. We settled on a design that has a spinning shaft that wraps up the straps, which tightens the straps around the air bladder and forces air from the bladder to the arm.

2.1.2 Initial testing of drill spinning shaft design

Figure 2.2: Prototype with a square rod spun by a drill

The first mock-up of the shaft spinning design used a 3/4″ (19 mm) square wooden dowel spun by a power drill. An adapter was printed to attach a 3/8″ (9.5 mm) aluminum rod to the shaft for mounting in the drill chuck. Plastic duct support straps [29] were stapled to the square rod. The system is shown in Figure 2.2 without the drill or anything being compressed. This design showed promise, but it was limited in the force it could apply by its square (non-rounded) shape. Once the rod was flat against the wood it would not turn any more and could not apply any more force.

The next iteration of this design replaced the square rod with a 3/8″ round wooden dowel, which was small enough to fit directly in the drill chuck without needing an adapter. A mount was 3D printed for each end of the rod to constrain the rod while it spun, and the straps were routed

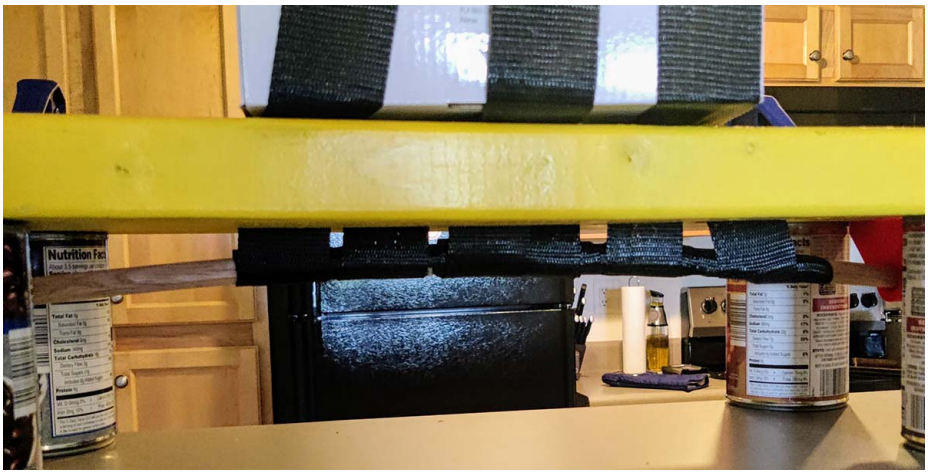


Figure 2.3: Deflection in prototype with $3/8''$ rod

through a gap between two 2x4 boards. Tests with this round-rod prototype showed that the shaft was able to apply some force to the objects, but it quickly warped as shown in Figure 2.3. The design struggled to crush a thin cardboard granola bar box, but easily squished a couch pillow.

(a) Bottom view (b) Front view

Figure 2.4: Prototype with $3/4''$ rod and spacers

To fix the deflection issue, the next prototype was built with a $3/4''$ wooden dowel and with 3D printed mounts between every plastic strap as shown in Figure 2.4. Testing with this mock-up was encouraging. It squeezed a couch cushion to the point that some stitches snapped, exploded a Thunder Stick [30], and was able to deploy a continuum arm prototype made of a Thunder stick and a larger inflatable cylinder [31] as shown in Figure 2.5. The wood on this setup started cracking

and splitting after a few tests, indicating that subsequent prototypes should use a stronger rod material such as metal. These tests suggested that the drill method would be fast enough to deploy in real-time as the user fell.

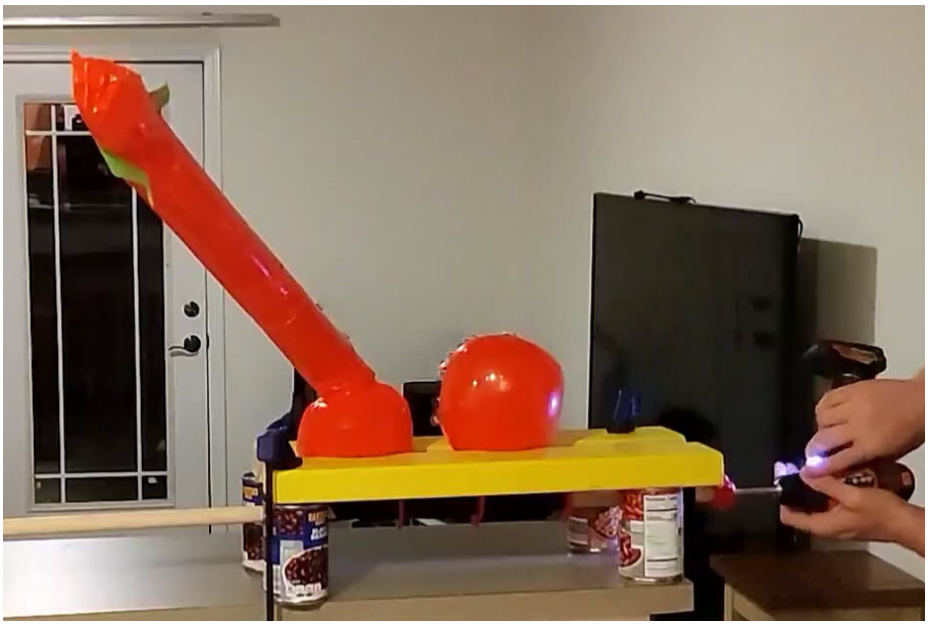


Figure 2.5: Small-scale prototype deploy test

The final prototype discussed above gave us confidence to move forward with the power drill and strap deployment method.

2.2 Arm and Bladder Construction

The fabric portion of the system was constructed out of Heat Sealable 200 Denier Oxford Nylon [32]. This material was chosen because in a similar inflatable continuum arm project [24] it could withstand the most pressure of all the materials tested. The fabric has a coating on one side similar to a very thin layer of hot glue. When two layers of the fabric are laid on top of each other so that each glue-layer side is facing each other, a soldering iron can be used to seal them together for a very tight seal. We also tested sealing the two sides with layers facing the same way (glue down for both, so that one glue layer was trying to seal to the non-glue side of the other piece) but this approach did not adhere nearly as well. We found that a temperature of 240°C paired with a large knife-style soldering iron tip worked best for sealing the sheets of fabric together.

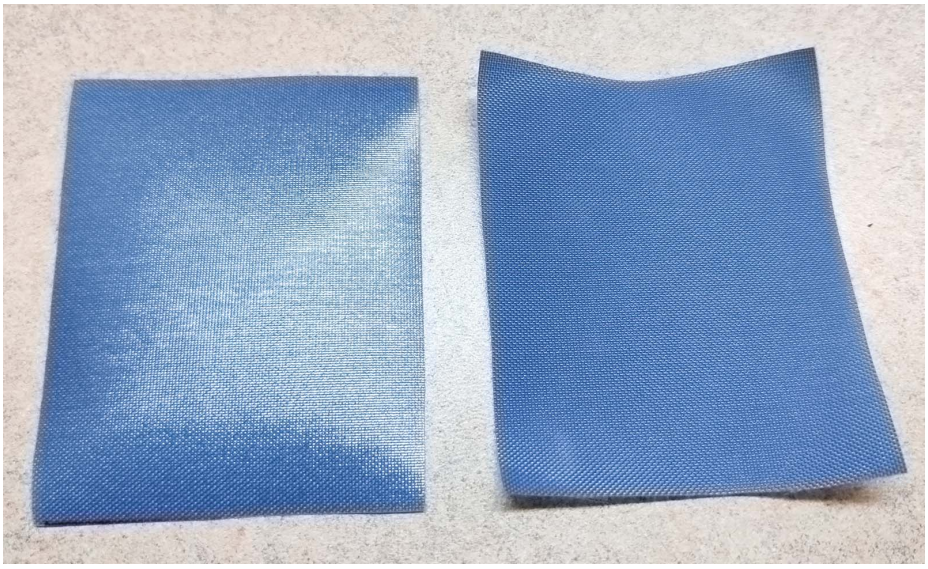


Figure 2.6: Adhesive side of the fabric (left) and uncoated side (right)

The fabric was sealed together by dragging a soldering iron pressed against the material at a very slow pace of about 3 seconds per millimeter. During testing, we found that moving the soldering iron too slowly can result in the iron melting through the material. Moving the soldering iron too quickly however can result in the material not forming a strong bond or any bond at all. Similarly, using the soldering iron at too high a temperature will melt the material before a seal can form, and too low a temperature requires an impractically slow speed or does not form a seal at all. We also found there is no increase in bond strength when using a large, flat soldering iron surface versus using the sharp part of the tip. If anything, the sharp part worked better because it pressed down with more force and sealed the arm more consistently.

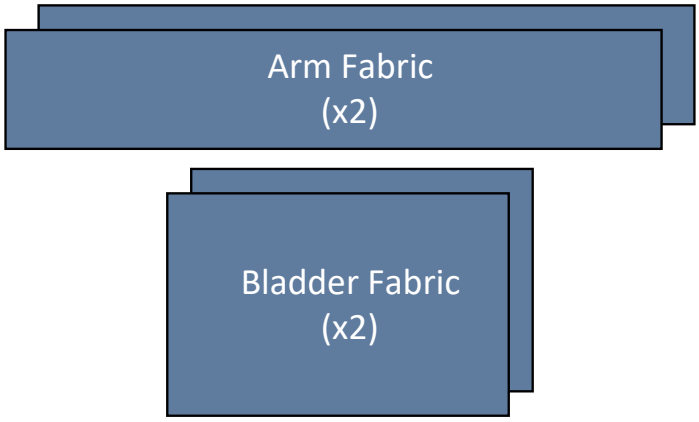


Figure 2.7: Identical fabric pieces cut out for the arm and bladder

The fabric assembly of the project was made of two parts - the arm and the air bladder, each of which was made from two identical cuts of fabric as shown in Figure 2.7. The arm was made from two pieces that were 60″ by 20″ (152 cm by 50 cm), and the bladder was made with two pieces that were 30″ by 20″ (76 cm by 51 cm). These sizes were chosen to leave approximately 1″ (25 mm) of space on any edge of the fabric for the two pieces to be sealed together.

The soldering iron was used to seal down one side of the arm, then the top, and then down the other long side. These first two sides had minimal issues with wrinkles, but on the second long side wrinkles occurred every 14″ (35 cm) or so due to the fabric not being secured very well while the first two sides were being sealed. Where the fabric did wrinkle, extra time was spent sealing up the wrinkle by pressing and holding the wide part of the soldering iron tip on the material and also making a few extra parallel lines that linked up with the main side lines. The act of sealing the system up by hand was not perfect - it proved easy for the top layer to be dragged across the bottom layer a fraction, and it took little of this to cause wrinkles. Luckily the methods for sealing up the wrinkles proved effective and there were minimal issues with air leakage throughout testing.

Figure 2.8: Air nozzle sealed to the arm

An air nozzle from an inflatable cylinder toy [31] was added to the arm as shown in Figure 2.8 to provide an easy way to increase or decrease the amount of air in the system between tests. The same soldering iron heat-sealing technique was used as on the rest of the arm, with the adhesive side of the fabric pressed against the tube fabric. The nozzle was attached before flipping the arm inside out.

The arm was flipped inside out so that the adhesive coated side was facing outwards, while

the air bladder had the adhesive coating on the inside. The arm slid inside the air bladder and was sealed in place with the same soldering iron sealing method, which required many parallel passes to ensure a strong bond. The assembled bladder and arm can be seen inflated in Figure 2.9.

There are several ways in which the manufacturing process could be improved. Researchers of fabric soft poly limbs used a CNC machine with a soldering iron attached to manufacture parts from nylon fabric [24]. This was important for consistency of that project, which required dozens of small pillows to be made identically. The size of our project was too large to fit on a desktop sized CNC machine; however, a common size for large CNC machines is $4'$ by $8'$ (122 cm by 244 cm) which would easily fit the arm herein. Using a CNC machine would require thorough testing of temperature, speed, pressure, and work-piece holding, but could result in a much more dimensionally accurate arm. Any way to apply consistent pressure on the soldering iron would be a huge improvement. We tested some 3D printed holders for the soldering iron and considered other approaches such as mounting the soldering iron stationary and moving the fabric beneath it like with a sewing machine. Eventually we decided that multiple passes manually was effective enough for creating our prototype. Other tools for applying heat could be explored, such as using a hair straightener or heat sealer machine [33].

Figure 2.9: First full inflation of the fabric arm and air bladder

2.2.1 Actuator Cable and Other Attachments

Figure 2.10: Components of the end of the arm

An important aspect of the arm's construction was the fusion of numerous 3D printed parts to the arm as shown in Figure 2.10. The arm shape was configured using four tendons, running longitudinally down the arm, and terminated at its tip. The tendons were equally spaced radially by 90 degrees. These tendons needed guides to keep them routed along the arm surface. Consequently, the arm had 32 individual cable guides printed in black PLA filament [34] in addition to 4 cable guides integrated in the pieces at each end of the arm. A 25″ (64 cm) long thin strap 3D printed with flexible black NinjaFlex filament [35] was passed through slots in the arm that held the cable guides in place.

A "crown" was located at the end of the arm where all four cables are attached, and this was also held in place with a flexible strap through strap guides. The crown weighed 122 grams before the cables were attached. The crown was originally printed in PLA, but that version broke during testing and was reprinted using semi-flexible TPU filament [36]. The 9 rows of cable guides were spaced every 5.6″ (14.2 cm), leaving 5″ (12.7 cm) between each cable guide. We added a 2.5″ (6.4 cm) section of spiral cable wrap [37] between each cable guide to improve the bending consistency of

the arm. Prior to the spiral cable wrap addition, the cables guides were not evenly distributed along the cable - several guides became pressed against each other, then only the last few guides close to the end would be the full 5 inch spacing apart. Figure 2.11 illustrates the relative dimensions and locations of the fabric system components.

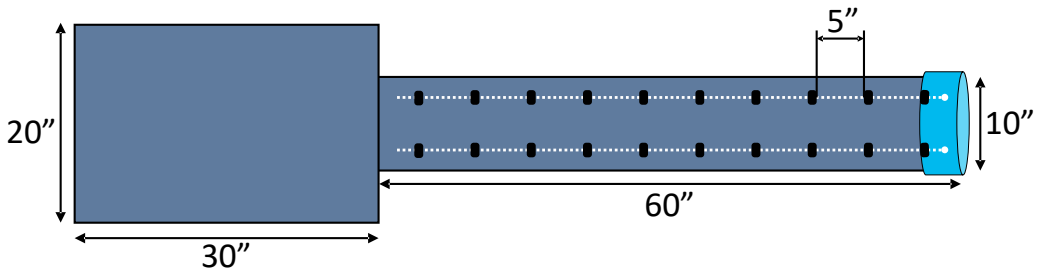


Figure 2.11: Relative dimensions and positions of fabric arm components

2.3 Mechanical and Hardware

Given the experience gained from the previously discussed initial testing, we proceeded to develop a full system integrated to a walker using the spinning shaft design. In this section, we will discuss the construction of the mechanical aspects of the systems integrating our novel inflation method, using a power drill spinning a shaft with straps to compress the air reservoir/bladder and inflate the arm. We attached the system to the frame of a walker and integrated it with the components that would change the shape of the arm within the time frame of a fall. The completed system is shown in Figure 2.12.

2.3.1 Selected Walker and Hardware

The system in this thesis was built around a Drive Medical 10210-1 Deluxe 2-Button Folding Walker with Wheels [38]. This walker was chosen because it was inexpensive, commonly used (more than 20,000 Amazon reviews), and not very stable compared to other walker styles. A Teensy 4.1 microcontroller [39] was used as the controller for the entire project. The other electronics components of the project are discussed in section 2.5 below.

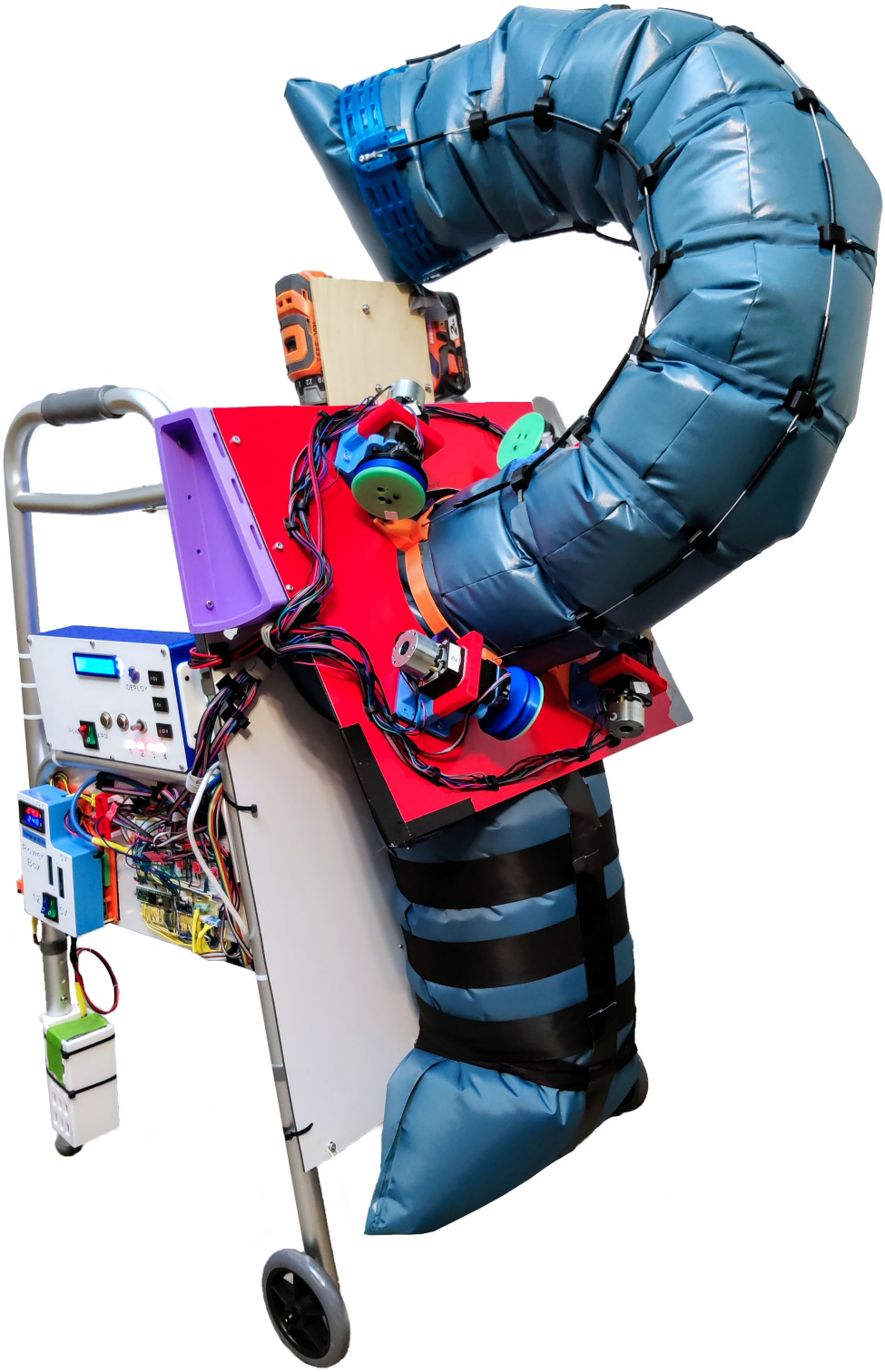


Figure 2.12: Front view of the whole system

2.4 Integration with Walker and Actuators

2.4.1 Frame

Figure 2.13: Cutting the slots for straps (left) and finished front plate on the walker (right)

The front plate was cut from 3.0 mm AluPOLY [40] to a size of 30″ by 20″ (75 cm by 50 cm) to fit on the front of the walker. A jigsaw was used to cut slots where the walker frame bowed outward, allowing the plate to sit flush against the front face of the walker. Oval slots for the nylon webbing straps to pass through were cut using a Dremel and a 3D printed cutting guide. The completed front plate is shown in Figure 2.13.

The edges of the AluPOLY slots were fairly rough and posed a risk of snagging the webbing straps. Figure 2.14 shows a 3D printed part with a smooth edge for the webbing straps to slide against. This part was installed on each side of the oval slot on the front plate. This greatly reduced the friction of the strap sliding against the edge.

The main spinning shaft was a 3/4″ (19 mm) square aluminum rod, which was larger than would fit in the drill chuck. The drill was therefore attached to the rod via a 3D printed adapter shown in Figure 2.15 that had an 8 mm hex bit [41] inside it. The hexagonal shank of the hex bit could be securely locked in the drill chuck, and the 8 mm head of the bit was fit snugly inside the printed plastic piece. Two M3 bolts were installed on the sides of the hex bit, so the force should be

Figure 2.14: Smooth guides for the webbing straps on the 3D printer (left) and partially installed on the front plate (right)

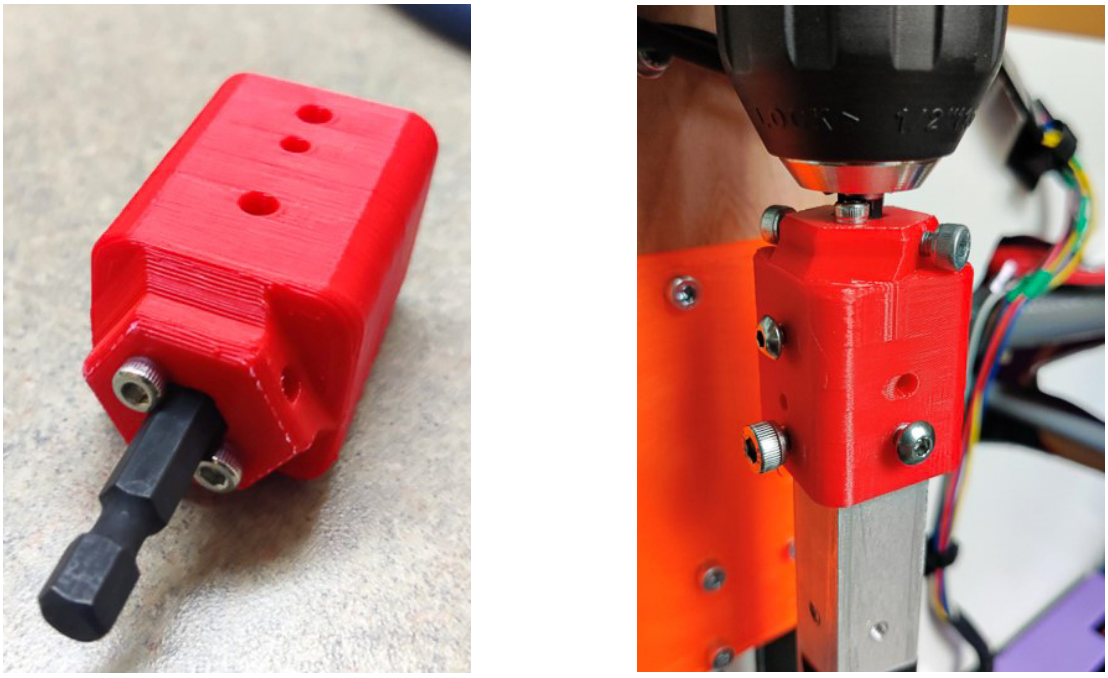


Figure 2.15: Adapter with embedded hex bit for connecting drill and square rod

translated from the metal of the 8 mm bit to the metal of the M3 bolts into the metal of the square tubing, without relying solely on the strength of the 3D printed part at any time. The adapter had M4 bolts that went through the outside of the adapter, through the square tubing, and into the inside of the adapter.

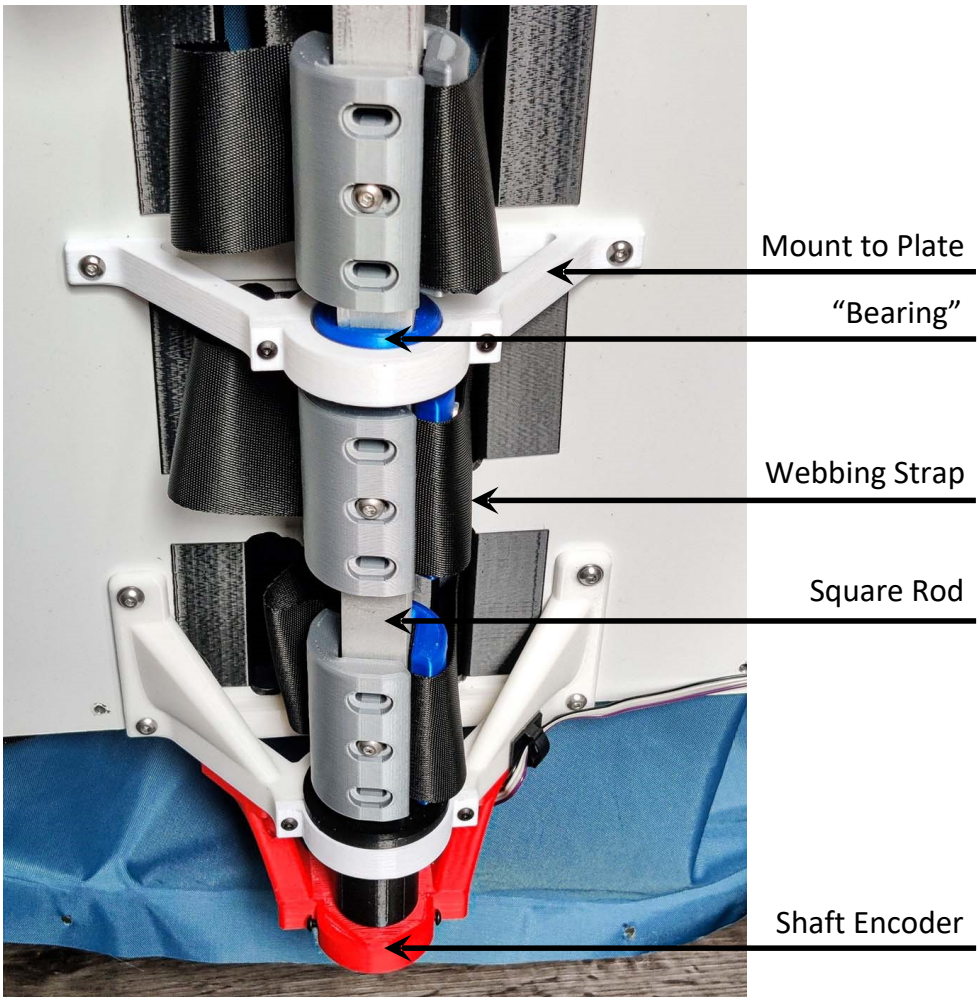


Figure 2.16: Spinning shaft components labeled

The next step was to integrate the straps with the drill and bladder/arm. Each webbing strap was held in place by two 3D printed pieces mounted on the square rod. These pieces were round on the outside, turning the square shaft effectively into a 40 mm (1.6″) round tube. M4 bolts were passed through the printed piece, webbing strap, and into a threaded hole in the aluminum square shaft. The two mounting holes in the webbing strap were made by pressing a hot soldering iron to the webbing strap, forming a 5 mm (0.2″) hole with melted edges that prevented fraying.

There was a "bearing" between every two webbing straps, which consisted of a 3D printed part that was square on the inside and round on the outside. This bearing helped distribute the forces on the rod and minimized the deflection of the rod. A small amount of SuperLube Synthetic Multi-Purpose Grease [42] was applied to each bearing after installation. The bottom bearing had a flat part on the bottom that held all the shaft weight when the shaft was vertical, as it would be during normal walker use. The bottom bearing also had an 8 mm rod sticking out from the end of the shaft into another AMT102 rotary encoder. These parts are shown and labeled in Figure 2.16.

The rod assembly was attached to the main plate by several 3D printed mounts. The mounts were attached to the main plate with M4 bolts and threaded holes. The mounts had a cap that fit around the shaft bearing with a 2 mm tolerance to allow for free spinning when accounting for small assembly errors.

The inflation system's method of actuation was based on a R860054 brushless drill [43] which spun a shaft that wrapped up the nylon webbing straps to squeeze air from the bladder into the arm. We chose to use a drill instead of a conventional geared DC motor for several reasons. First, drills have a gearbox with two speed settings (1 and 2) built in, with more torque but lower speed available on speed 1 than speed 2. Drills have an adjustable clutch for changing the maximum torque output, which helped us protect (not burst) the air bladder or arm during initial testing. Drills have overcurrent protection and stall protection built in, along with a battery system that can provide the necessary instantaneous current for the drill. Drills have a large metal chuck on the end that can securely hold many different shaft shapes and sizes. All these features are advantages that a drill has over creating a custom solution with a basic DC motor and gearbox. The drill solution worked well for our system, in which torque and speed were more important than precision or accuracy. The drill was mounted to a 3D printed plate with six zip ties. The drill was programmed to deploy spinning in the counterclockwise direction so the forces pressed the drill against the 3D printed plate instead of pulling against the zip ties.

A rear view of the final overall system is shown in Figure 2.17. Along with the mechanical components discussed previously, this figure shows several boxes which hold electrical components of the system which will be discussed in the next section.

Figure 2.17: Overview of walker rear side

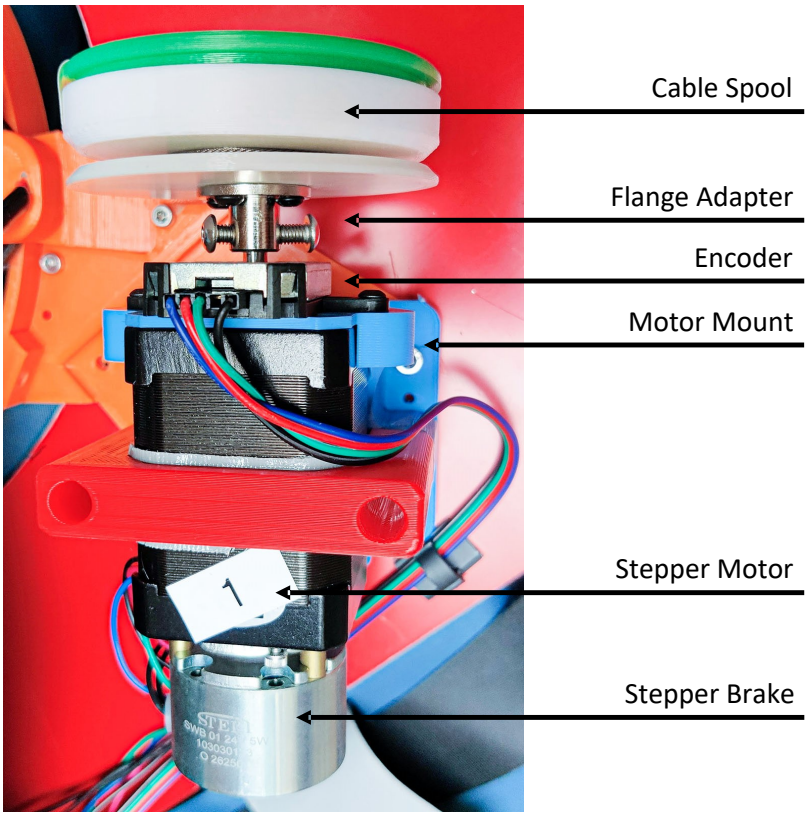


Figure 2.18: The completed motor assembly for one arm cable

2.4.2 Arm Motors and Spools

With the airbag deployment method completed, we next developed hardware that would control the curvature of the continuum arm when the system was deployed. Each steel cable tendon on the arm was stored in a cable reel or spool that was on a stepper motor shaft. When the cable was pulled out to the desired location, the motor would apply a braking force to stop the cable from being pulled out any further.

The motor assembly was constructed as shown in figure 2.18. A dual-shaft NEMA 17 stepper motor [44] was attached to a 3D printed mount. An AMT102-V rotary encoder [45] was attached to the front of the same mount. An electromagnetic stepper motor brake [46] was mounted to the rear side of the stepper motor. A stainless steel flange adapter [47] was connected to the motor shaft, and the cable spool assembly was mounted to the flange adapter. The cable was threaded through a hole on the inside of the motor spool and locked in place by a M4 bolt, then the excess length of cable could be held on the outside of the spool (between the green piece and white piece in figure 2.18). The steel motor flange was manually threaded to use M4 bolts to provide better holding strength than the M3 threads it originally was supplied with.

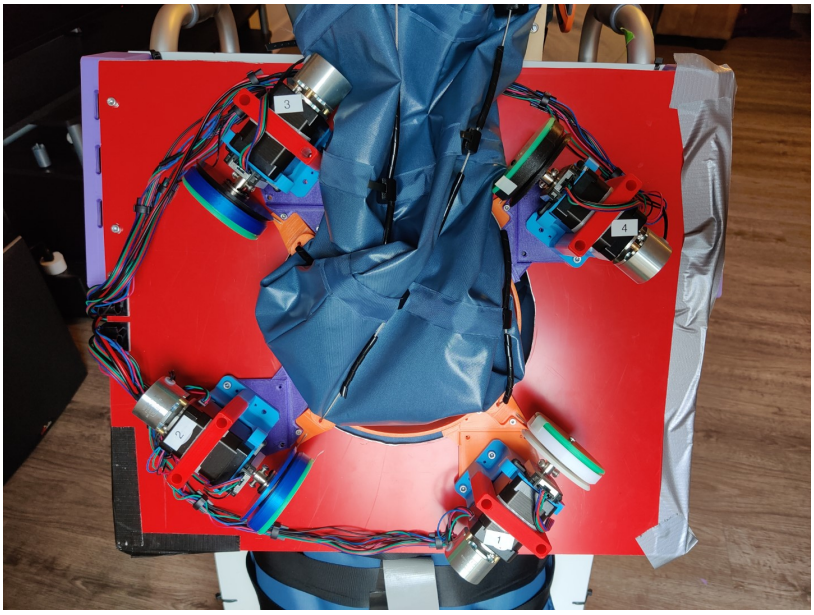


Figure 2.19: All motor assemblies mounted on the walker

This motor and spool assembly was then mounted to the outside of an arm ring using M4 bolts. The arm ring had an internal path that guides the cable from vertical to horizontal. The arm

ring mounted to the angled plate that was offset from the walker to provide an adequate bending radius for the fabric of the arm. Holes were drilled into the angled plate and threaded for M4 bolts to reduce the need for nuts, however if any holes were accidentally stripped a M4 nut could be attached to the back of the bolt to hold it securely. The angled plate was originally held in place by an adjustable hinge, but once a distance from the walker and angle was finalized a stronger, non-adjustable mounting piece was printed and installed. The final assembly is shown in Figure 2.19.

2.5 Electrical Design and Build

In this section we describe the electrical aspects of the system. The prototype developed in this project was completely untethered - everything ran off batteries, and all required electronics were mounted on the walker frame. Several of the electrical systems can be seen mounted to the outside right of the walker frame in Figure 2.20 including the control panel (top), voltage regulation (bottom left), motor drivers, and microcontroller (bottom right). The key components of the electronics aspect of the system are listed in table 2.1. In addition to the items listed in the table, numerous standard electronic components were used including push-buttons, switches, assorted connectors (XT-60, JST-XH, Dupont, 5.5 mm barrel jack), 22-AWG solid core wire, screw terminals, and breadboards.

Part	**Specs/Details**	**Quantity**
Teensy 4.1	Teensy 4.1 Development Board[39]	1
GPIO expander	Adafruit AW9523 [48]	1
Battery	Floureon 6S 4500mAh 45C [49]	1
Battery display	Voltage+current display meter [50]	1
5V regulator	Pololu D36V50F5 5V 5.5A [51]	1
12V regulator	Pololu D24V150F12 12V 15A [52]	1
Fuses	Inline fuse holders + fuse	2
Motor drivers	DRV8825 with heat sink	4
Stepper motors	Dual shaft NEMA 17 2.1A [44]	4
Motor brakes	DC electromagnetic brake 24V [46]	4
Drill	Ridgid R860054 drill/driver [43]	1
Digital potentiometer	HiLetgo X9C104 40-100 kΩ [53]	2
Relays	Mechanical relay modules	6
9-DOF IMU	Sparkfun ICM-20948 [54]	1
Level shifters	TXS0108E 3.3V 5V bi-directional [55]	2
Knob encoder	Grayhill 61C optical encoder [56]	1
Rotary encoder	AMT102-V [45]	5
LCD display	16x2 LCD with I2C/SPI adapter	1
LED strip	5V WS2812 with 4 LEDs	1

Table 2.1: Major electronic components

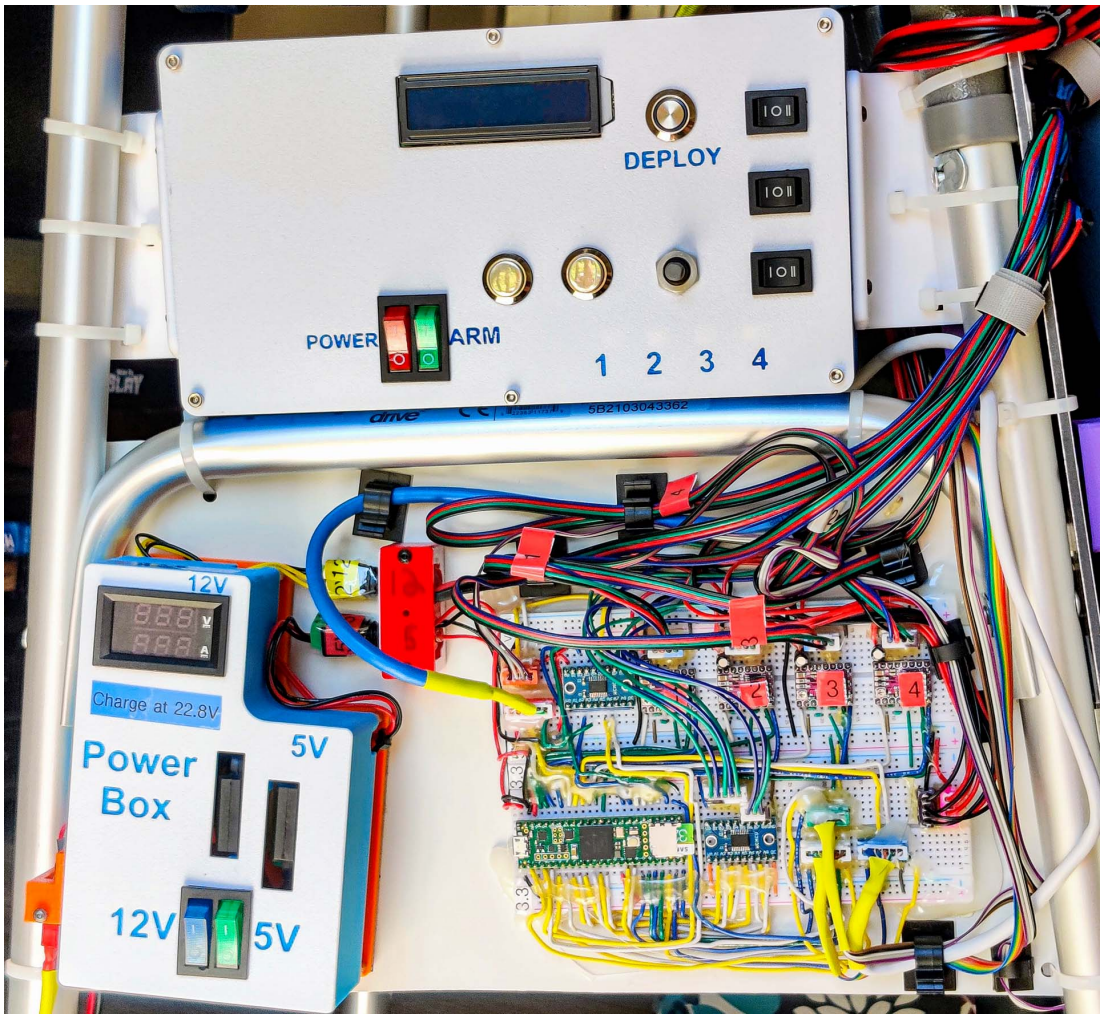


Figure 2.20: Main electrical control systems for the project on the right side of the walker frame

2.5.1 Voltage Regulation

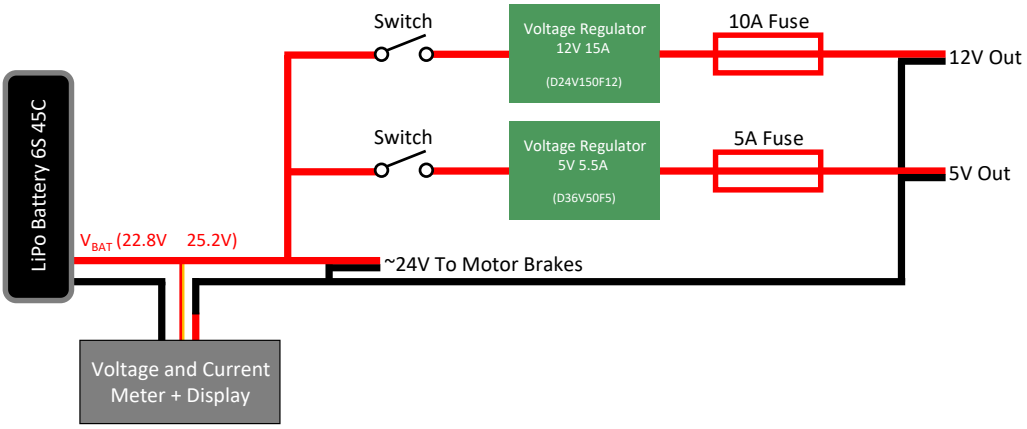


Figure 2.21: Voltage regulation circuit diagram

The system was powered by a 6S 22.2V 4500mAh 45C LiPo Battery [49]. Lithium-ion Polymer (LiPo) batteries have a relatively large voltage swing from fully charged to empty. In this case the fully charged voltage was around 25.2V and the fully discharged voltage was around 22.8V. Figure 2.21 shows the voltage regulation circuitry for the project. A voltage and current display [50] was placed in line with the ground connection of the battery and was powered by the positive battery connection. One connection from the battery went directly to the electromagnetic motor brakes switch since they required 24V ±10% to run. A Pololu D24V150F12 12V 15A voltage regulator stepped down the battery voltage to 12V for the stepper motor drivers. A Pololu D36V50F5 5V 5.5A voltage regulator provided a steady 5V for the Teensy 4.1 and a variety of other components that require 5V to function. Each output from the voltage regulator had a fuse for overcurrent protection and a switch for easy control built in.

2.5.2 Microcontroller

A Teensy 4.1 Development Board was the sole microcontroller for the system. The Teensy 4.1 board was based on the ARM Cortex-M7 processor running at 600 MHz. It had 41 I/O pins easily accessible on the sides of the board, 27 of which were PWM capable and all of which have external interrupt capability. Figure 2.22 shows the function of each pin on the Teensy board in this project. Most pin locations were selected for convenience of wiring, since all digital pins on the Teensy 4.1 can be inputs, outputs, and hardware interrupts.

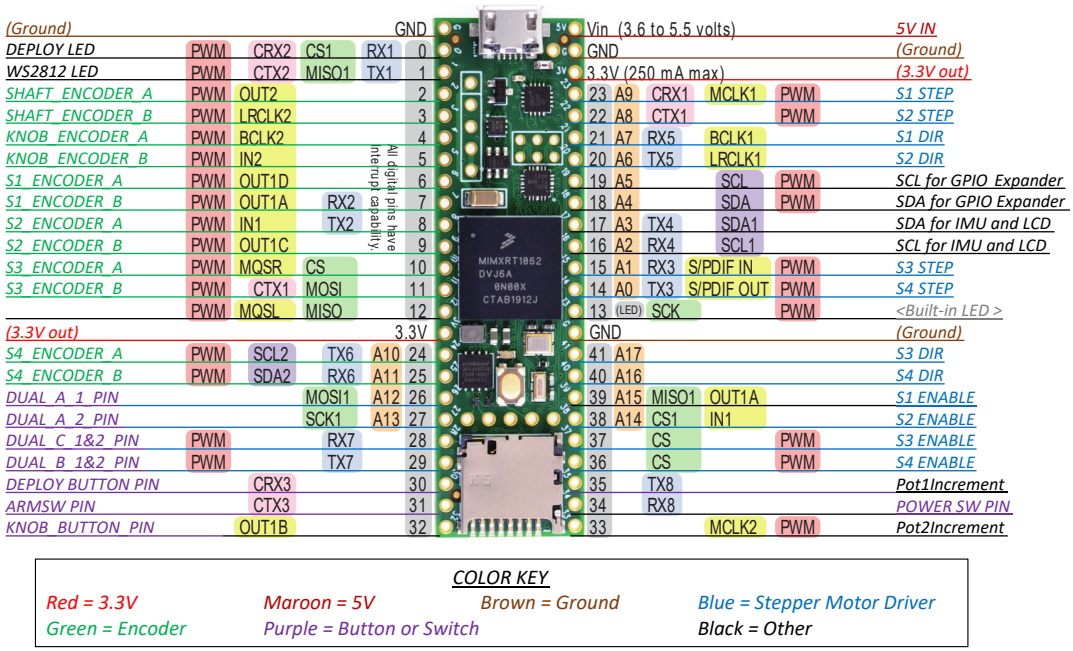


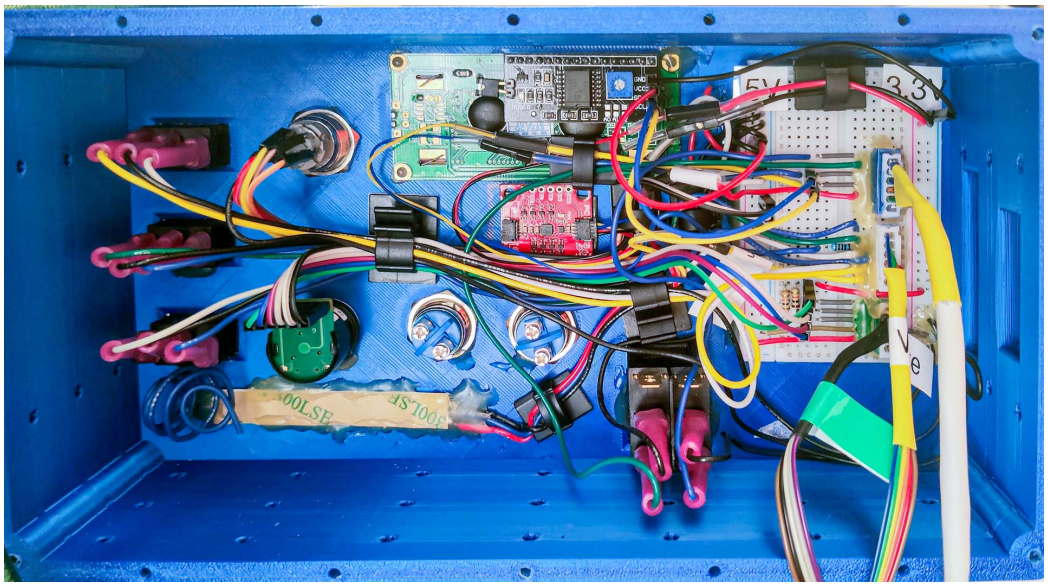
Figure 2.22: Overall microcontroller pinout diagram

Two separate I2C buses were used because the project had I2C devices in two separate physical locations on the walker. Initial testing used ribbon cables for the I2C wires, but issues with noise on the signal arose. I2C is a protocol designed for short-distance communication, typically with wires just a few inches long. The cables in our project were several feet long and travelled by some high-noise components such as motor drivers and motor chips. The problem was resolved by using CAT5e cable for the I2C connections. CAT5e cable is a standard cable type used in networking, and it uses twisted pairs of wires to help cancel out electromagnetic interference from external sources.

2.5.3 Control Panel

The control panel for the system is shown in Figure 2.23. The three switches were used to switch to a mode that controls the drill, stepper motors, or motor brakes. The rotary knob was used to turn the various motors in the specified direction (clockwise or counterclockwise). The green switch labeled “ARM” was used to arm the system in preparation for deploying. Some items on the control panel were not wired up, such as the silver buttons in the middle. These items were included in the initial control panel design in case a need for more inputs arose during the development of the system.

(a) Front view



(b) Rear view

Figure 2.23: Electrical control panel

2.5.4 Drill Trigger Replacement

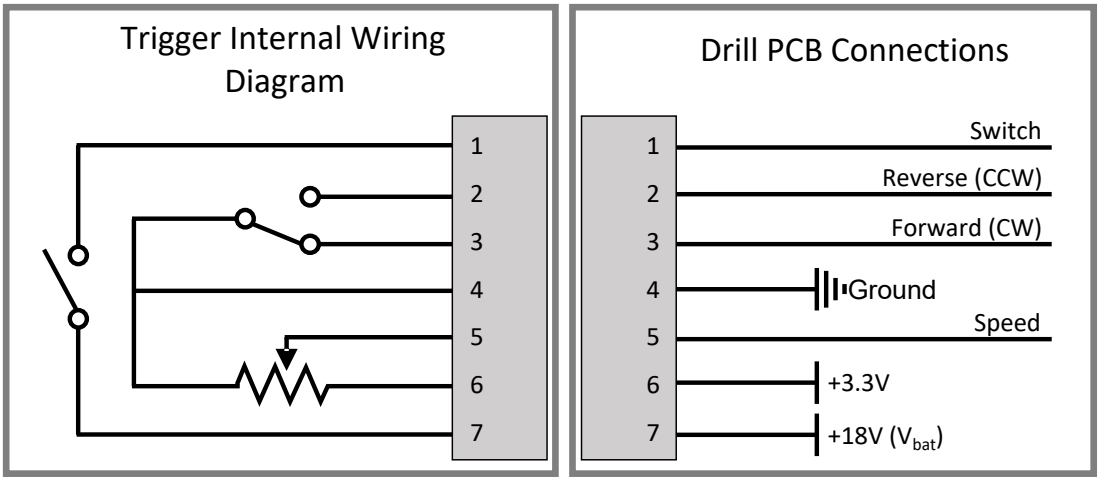


Figure 2.24: The wiring diagram of a power drill trigger. A SPDT switch selects the forward or reverse direction, a potentiometer controls the speed, and a SPST switch signals if the trigger is pulled or not.

The trigger on the drill was replaced with our own electrical components so that we could control the drill using the microcontroller. The trigger was connected to the drill with a 7-pin connector as shown in Figure 2.24. The leftmost switch on the diagram is single pole single throw (SPST) and is triggered when the trigger is slightly pulled in. The single pole double throw (SPDT) switch selects the direction of forward (clockwise, CW) or reverse (counterclockwise, CCW), and the potentiometer value controls the speed to move the drill faster as the trigger is pulled in further. For our drill, the potentiometer was 300 Ω before the trigger was pressed in, and was 32.5 kΩ when the trigger was pulled completely.

The direction switch and trigger switch were replaced with mechanical relays, and the potentiometer for speed control was replaced with a pair of digital potentiometers. These replacements allowed us to control the drill with the microcontroller.

The XC1902 digital potentiometers consisted of an array of 99 resistors, wiper switches, a control selection, and memory [57]. The wiper changed when the increment (INC) input was toggled with the corresponding direction input for the up down pin (U/D). The memory could store the last position of the potentiometer so that it returned to that position on the following startup. A block diagram of the digital potentiometer (Figure 2.26) shows the resistor array within the chip. There was a library written to control these digital potentiometers that had convenient

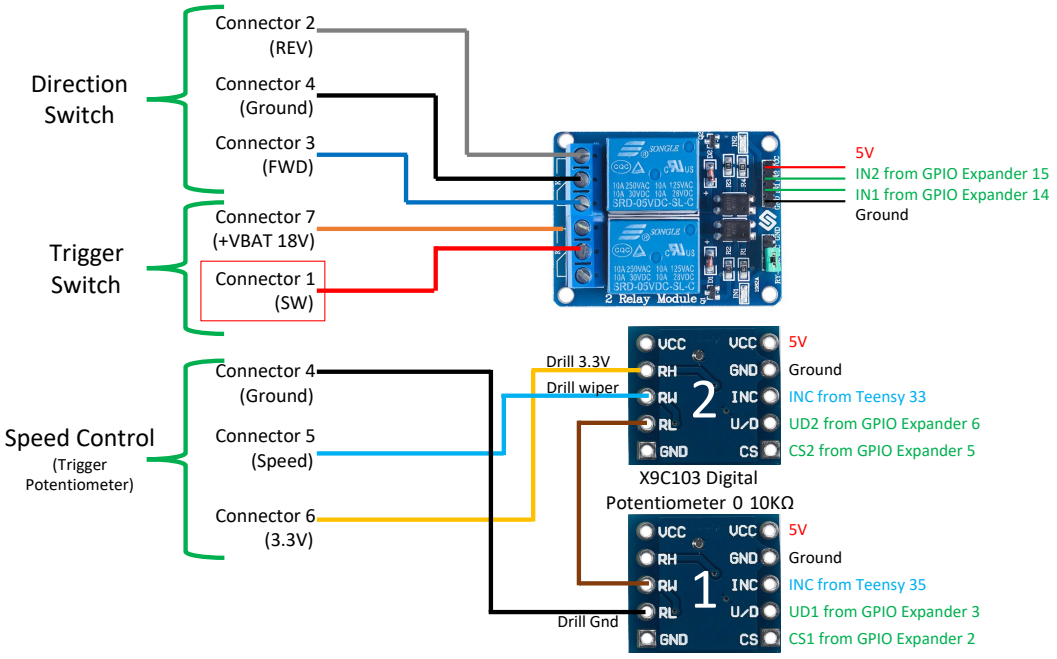


Figure 2.25: Drill trigger replacement circuit diagram

functions for increasing, decreasing, or setting the value (0-99) of the potentiometer [58]. Each digital potentiometer resistance could only go up to 10 kΩ, but the drill trigger needed around 20 kΩ to run at full speed. Two digital potentiometers were placed in series to combine their resistance to reach the required 20 kΩ for full drill speed. The connections for the replacement drill trigger system are shown in Figure 2.25.

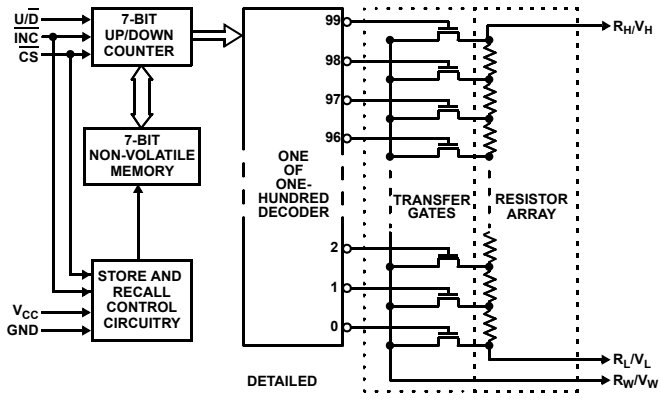


Figure 2.26: X9C102 digital potentiometer block diagram

The Adafruit AW9523 GPIO expander provided 16 I/O pins through a single I2C connec-

tion. It was installed inside the drill control box along with six relays and two X9C102 digital potentiometers. Figure 2.27 shows the functionality of each pin as they were used in this project. We used the Adafruit AW9523 library for Arduino to interface with the expander over I2C [59].

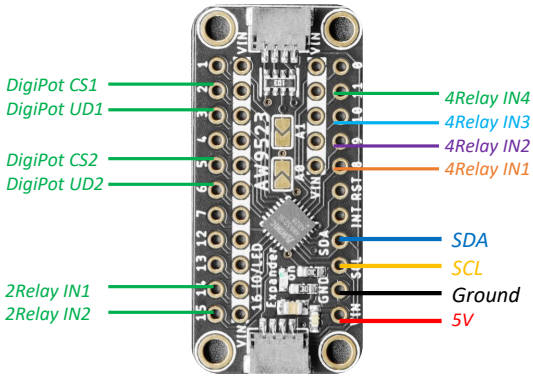


Figure 2.27: GPIO expander circuit diagram

The increment pin for the digital potentiometer was wired directly to a Teensy pin instead of a pin on the GPIO expander. Each digital read or write on the GPIO expander used a I2C command, which took more than 1ms per command. The potentiometers changed their resistance up by one step every time the increment pin was set high and low, so changing both potentiometers from spot 0 to 99 took 600 ms on the GPIO expander. The same action took 8 ms using a Teensy pin directly for the increment signal.

2.5.5 Brakes

Each stepper motor had a 24V electromagnetic brake [46] attached to the rear of it. The brake increased the holding torque and stopping torque of the motor so that it could stop the cable spool at the desired location as the arm inflated. Each brake was controlled by a mechanical relay as shown in figure 2.28. The brakes were engaged (applying braking force) when no voltage is applied, and released when 24V ±10% was applied. The set of four brakes drew 800mA at 24V when they were enabled.

The components described above (relays for motor brakes, relays and digital potentiometers for drill control, and GPIO expander) can be seen in Figure 2.29. The box was 3D printed with mounting holes for the components and cutouts for the wires to pass through. There is a switch on the lid of the control box for switching the 24V directly from the LiPo battery.

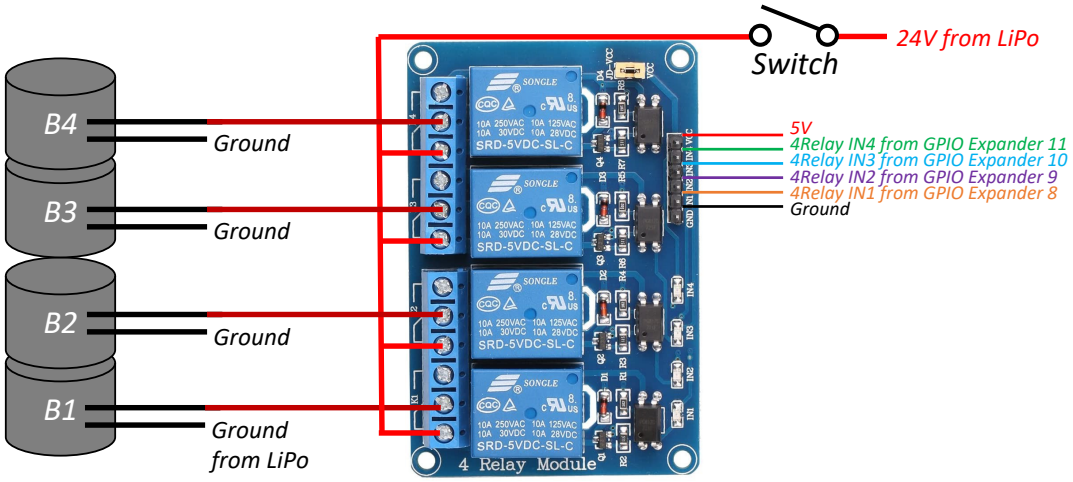


Figure 2.28: Electromagnetic motor brakes circuit diagram

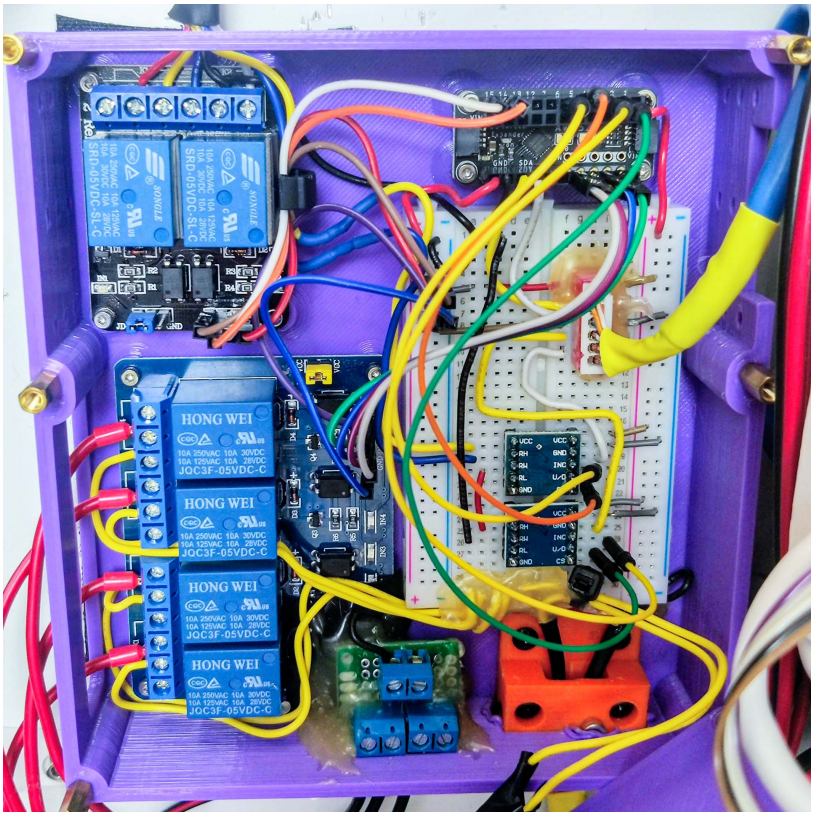


Figure 2.29: Inside the electrical control box with relays and digital potentiometers to interface with the brakes and drill.

2.5.6 Motor Drivers

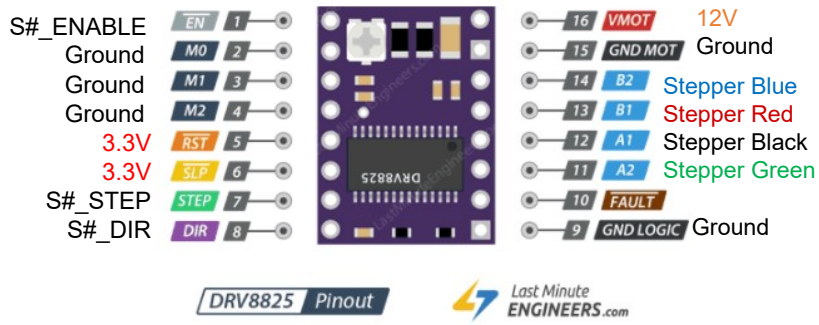


Figure 2.30: DRV8825 motor driver pinout for our system

The DRV8825 stepper motor drivers were wired according to the DRV8825 tutorial at the Last Minute Engineers website [60]. Figure 2.30 shows the motor driver wiring for our system using 12V as the motor voltage, 3.3V as the logic voltage, and the correct colored stepper motor wire in each output for our specific stepper motors. The microstep selection pins M_0, M_1, and M_2 are all grounded so the motor driver is in full-step mode and is not microstepping. Microstepping is used to provide greater accuracy (steps per revolution), but the maximum speed and holding torque are both reduced in microstep mode. The potentiometer on the motor driver was adjusted following the "Current limiting, Method 1" on the website to set the reference voltage (V_{ref} to 1.05V, corresponding to a current limit of 2.1A. The connections labeled *S#_ENABLE*, *S#_STEP*, and *S#_DIR* went directly to the Teensy microcontroller pins for each stepper motor (1-4).

2.5.7 Encoders

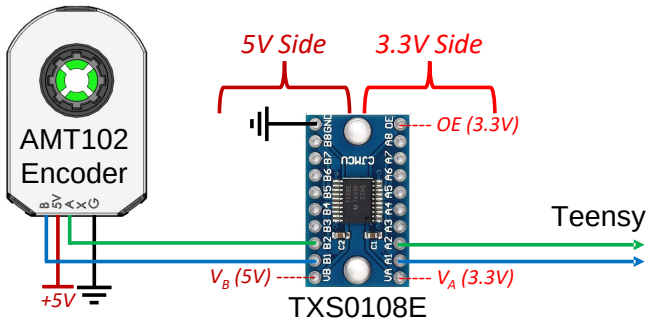


Figure 2.31: Encoder and level shifter circuit diagram

The AMT102-V quadrature incremental encoders support 16 different resolutions, from 48 pulse per revolution (PPR) to 2048 PPR. We determined that 48 PPR would be adequate for our system and would place the least burden on our microcontroller, and the appropriate DIP switches were set before the encoders were mounted in their respective locations. The encoders were powered with 5V (V_{DD}), resulting in an output high level of $V_{DD} - 0.8 = 4.2V$ according to the datasheet. However, the Teensy 4.1 microcontroller can only handle up to 3.3V on its input pins, so the output (A and B) from each encoder was put through a TXS0108E level shifter before going to the Teensy as shown in Figure 2.31. The TXS0108E has a voltage range of 1.2V-3.6V for the VCC_A side, and a voltage range of 1.65V-5.5V for the VCC_B side. For our system, we used 3.3V on the VCC_A side and 5V on the VCC_B side.

2.6 Software

In this section, the system's control logic and flow is discussed. The full code for the project can be found in the GitHub repository [61], and a few functions are detailed in Appendix B.

The Teensy 4.1 microcontroller was programmed using the Arduino IDE 1.8.15 in a Windows 10 environment. Arduino code is similar to C or C++, but with many extra libraries that can be installed for interfacing with various hardware. Arduino programs have a *setup()* function that runs once when the microcontroller is powered up or reset, and has a *loop()* function that loops infinitely. Our system was set up as a finite-state machine (FSM) that can only be in one state at a given time.

The code checks the value of the control panel inputs and determines which state to run. For example, if the lower right switch on the control panel is toggled, the system goes into "Drill Control" mode where turning the rotary knob turns the drill in the same direction. Similarly, the middle right switch enters "Stepper Control" modes where each click of the rotary knob turns a single stepper 90° if the switch is in position 1, or it turns all four motors 90° if the switch is in position 2. The LCD display shows the current mode in the top row and the current direction of the motors in the bottom row.

An important part of any airbag system is that it must only deploy when necessary and expected. Our system only arms the system if the "ARM" switch is toggled on and all other switches are off. The system also requires the ARM switch to be toggled off upon microcontroller reset, to prevent the case where the walker is turned on while laying on the ground and the airbag would immediately attempt to deploy.

2.6.1 System Arming

When the ARM switch was toggled on, the system became ready to deploy. Several things occurred when the ARM switch was flipped to prepare for deploying:

- All the stepper motors were turned on so they held the cables in place. They could still be controlled with the "tuning" features on the control panel or could be manually forced by hand.
- Electromagnetic brakes were released.
- The system saved the IMU value at the time of system arming as the "flat ground" value.

Further IMU measurements would be compared to this initial value.

- The drill was turned slightly to the deploy direction (CCW). When the drill changed direction from CCW to CW it took approximately 500ms, so turning it slightly in the correct direction ensured it did not take that extra time to deploy.

After these initial steps the system was considered armed. The IMU monitored changes in tilt, and if the walker tilted beyond a threshold in a certain direction (left, right, or forward) the arm would deploy in that direction. The deploy button on the control panel was used in initial testing before the IMU functionality was integrated with the system.

2.6.2 Deploy

Tendon-driven continuum robots typically bend by shortening tendons on the side the robot is meant to bend towards. Our system works on a similar principle, except our tendon (steel cable) is shortened by applying the brake at a certain time. The motor assemblies are numbered in the code according to Figure 2.32. To bend the arm to the right side of the image, the cables on assemblies 1 and 4 would need to be shorter than cables 2 and 3.

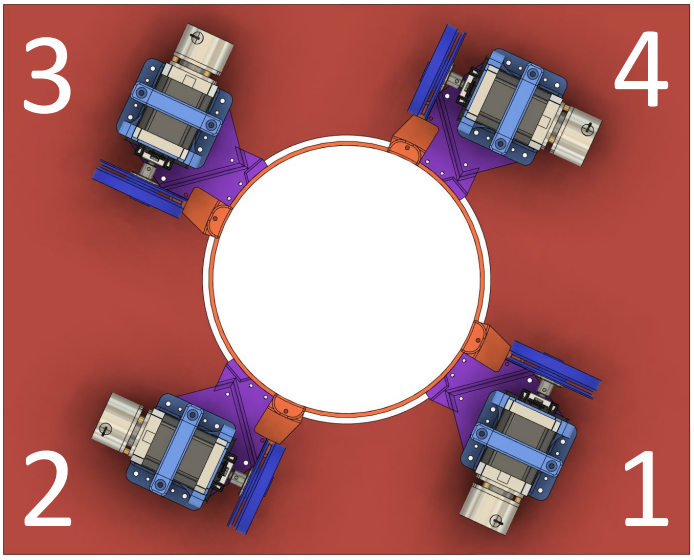


Figure 2.32: Cable assemblies as referenced in the software

The fall directions of left, right, and forward were referenced in the software as being from the perspective of the user, i.e. cables 1 and 4 are on the left side and cables 2 and 3 are on the

right side. Table 2.2 shows the encoder spot at which the brake and motor driver were engaged to stop its respective cable for each bending location.

Bend Direction	Encoder 1	Encoder 2	Encoder 3	Encoder 4
Left	50	1500	500	50
Right	1500	50	10	900
Forward Up	1200	1200	50	50
Forward Down	150	10	900	900

Table 2.2: Encoder values to engage brakes for different arm configurations

When the IMU tilt value exceeded the threshold in a particular direction, the system deployed. The deploy function was set up as a blocking function that would run completely without interruption, even if one of the control switches was accidentally toggled during the fall. Deploying followed these steps:

1. All stepper motors drivers were disabled and brakes disengaged.
2. The drill spun at full speed CCW, wrapping the straps and forcing air from the bladder into the arm.
3. Once an encoder was in the desired end location, the motor driver was enabled and the brake was engaged to stop the cable spool and hold the cable in place. This occurred separately for each of the four cable systems.
4. The drill was stopped after 700 ms or 3.5 full revolutions, whichever occurred first.
5. The deploy was considered completed after 2000 ms (2 seconds), and the system completed a few steps to "un-deploy" or relax the system:
   (a) The motor drivers were disabled and the motor brakes were disengaged.
   (b) The drill spun in the opposite direction (CW) at a slower speed for approximately 1 revolution to release some pressure on the arm.
   (c) A CSV log file was saved to the micro SD card about encoder, IMU, and digital I/O values throughout the deploy.
   (d) The system waited for the ARM switch to be reset.

The system would not deploy again until the ARM switch was toggled off and then back on, to ensure the system never tried to deploy consecutively.

The system needed to be manually reset after each deployment, similar to vehicle airbags. The resetting process included:

- Using the control panel to turn the drill to its starting position where the straps were not applying pressure to the bladder.
- Adding a small amount of air to replace that which had leaked out during deploying.
- Pushing air from the arm back into the air bladder.
- Ensuring all cables were on their cable spools and not tangled anywhere.
- Inspecting the system for damage.
- Rewinding each cable spool to the starting position where every spacer is touching the cable guide before and after it.

Chapter 3

Results Using the Prototype

When the system assembly and programming were complete, we evaluated the system's functionality through dozens of individual tests. This section describes the testing process and discusses some improvements we made to the hardware and data collection process during testing. Then, a complete set of final results is presented, including comparisons between baseline falls and deploy falls in each direction (forward, left, and right).

3.1 Preliminary Deploy Tests

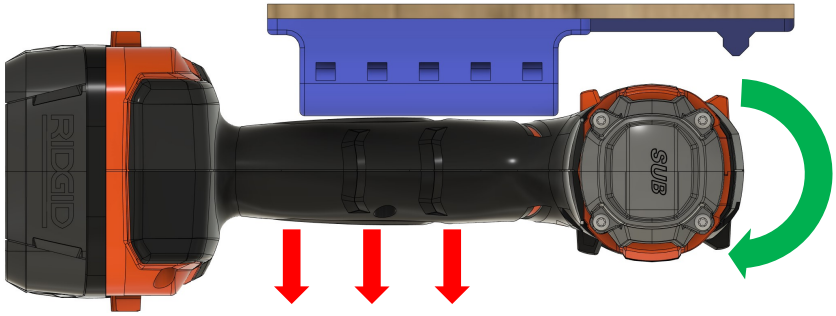
3.1.1 Speed and Clutch Settings

The first several tests were conducted to see what drill settings would be needed to fully deploy the arm, i.e. fully force air from the air bladder into the continuum arm. During these tests the motors and cable spools were allowed to spin freely, so the arm should have just deployed forward with no bending. Before testing, we anticipated the drill would be powerful enough to deploy the system on the speed 2 setting (faster speed, lower torque) and perhaps with a clutch setting below maximum. We found with the clutch set to any setting below maximum, the drill stopped spinning very early on, before almost any air could be forced into the arm. With the clutch set to maximum and speed set to 2 the system worked slightly better, but still only partially pushed air into the arm. This result can be seen in Figure 3.1 - the drill stopped spinning before the arm was inflated to a useful amount.

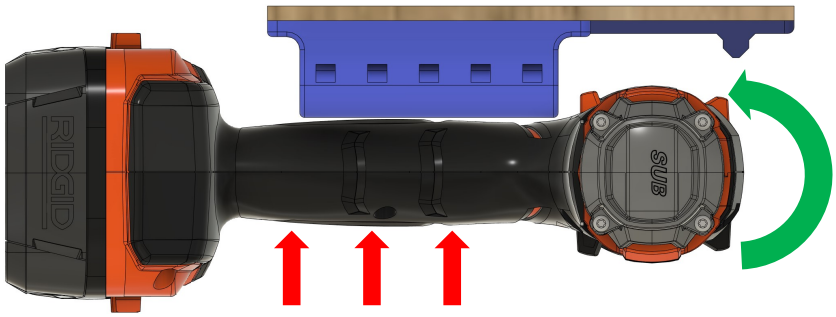
Figure 3.1: Deploy test with speed set to 2 and clutch on maximum. There is still air in the bladder that should be forced into the arm, but the drill stopped spinning before the arm was sufficiently pressurized.

The first deploy test that sufficiently pressurized the arm was done with the drill set to speed 1 (lower speed, higher torque) and the clutch set to maximum. This test inflated the arm rapidly and completely. However, this test broke several components of the system. In these first tests, the drill was set to spin in the clockwise direction while deploying. The clockwise force on the spinning shaft results in a counterclockwise force on the drill handle, which pulls on the zip ties in the setup. An illustration of these forces is shown in Figure 3.2.

During this successful deploy test at maximum torque settings, the drill snapped all three zip ties and tore loose several of the trigger wires from the digital potentiometers and relays in the system. After repairs, we reprogrammed the system to deploy in the counterclockwise direction so the handle force was pushing into the wooden mounting plate instead of pulling on the zip ties. We also increased the number of zip ties holding the drill in place from 3 to 7 for good measure.



(a) Forces when drill spins clockwise

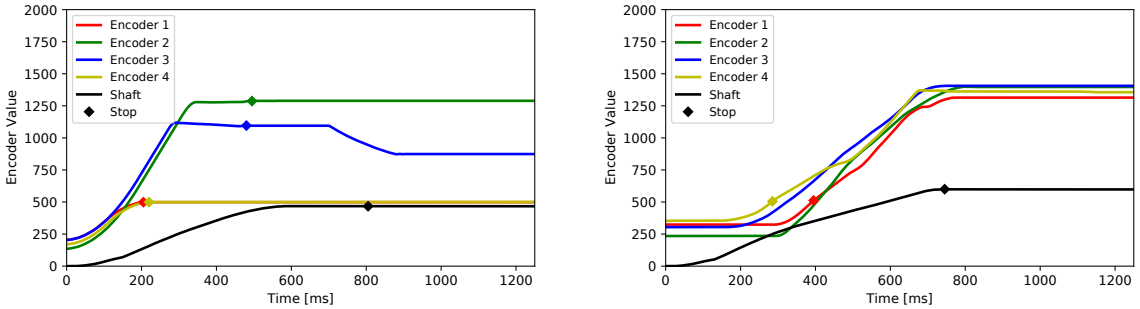


(b) Forces when drill spins counter-clockwise

Figure 3.2: Force diagram of the drill when it spins clockwise (top) vs counterclockwise (bottom). Later tests were run with the drill spinning counterclockwise so that the force pushed the handle into the wood instead of pulling on the thin plastic zip ties.

3.1.2 Cable Deploy Method

For this thesis, "driving" a motor to a position refers to using the driver and step signals to move the motor a certain number of steps. This is a simple method of actively moving the motor to a desired position. The other way the motors were moved in the system was by the steel cables pulling on the cable spools attached to the motors as the airbag deployed, which spun the motors. When a motor driver is enabled it applies the maximum amount of holding torque to its motor. Some motor drivers have a programmatically variable holding torque, but our DRV8825 do not have that feature. Initial testing showed that the motors had significantly less braking force when the driver was enabled while the cable was moving the motor, so we anticipated that driving them to position and then stopping them would be more effective than enabling the driver while the motor was already being pulled by the cable.



(a) Driving motors to position

(b) Motors free spinning

Figure 3.3: Deploying by driving the motors to position (left) vs letting the arm pull them to position (right). This shows the motors can move faster than the arm pulls them out, so either deploy method is an option.

We performed several tests using different methods to stop the cables at their desired positions. Figure 3.3 shows a comparison between the two methods across the entire 2 seconds of the arm deployment. In the left image, the motors are actively driven to their goal position as soon as the system deploys (walker detected as falling, or the deploy button was pressed) and then stop there. In the right image, the motor drivers are not enabled until the steel cable pulls the motor the same number of revolutions. The slope of the left graph (motors driving) is steeper than the slope on the right graph (pulled out), showing that the motors are able to move faster than the arm pulls the cables.

The diamonds on the images show where the motor driver is enabled and applying the

holding torque. The motors should have stopped at or soon after the motor driver was enabled. However, in the right image the motors continue to be dragged out by the arm deploying until they are all a similar length meaning the arm is straight.

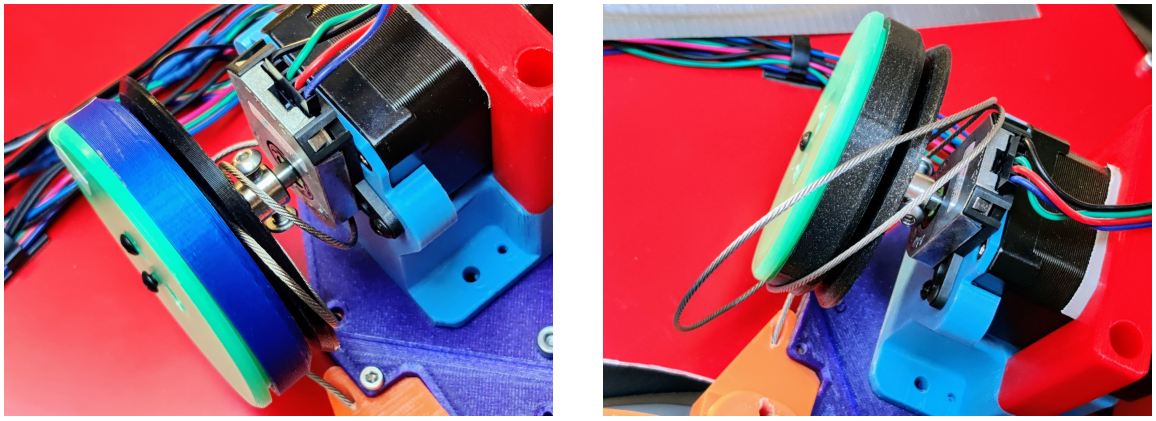


Figure 3.4: Motor cables tangled and wrapped up after driving the motors to position at max speed

We quickly discovered a serious issue with the method of actively driving the motors to their positions. The steel cables tended to unspool faster than the arm pulled them out, so in many cases they ended up tangled or wrapped around the cable spool or motor flange. This resulted in the cables bending the arm earlier than desired because the cable would be tangled and could not spin the motor at all. This issue is shown in video screenshots in Figure 3.4.

Next, we tried a hybrid solution where the motors were driven to their position but only started driving after the arm pulled them a few steps. We hoped that by delaying the motor starting time the motors would drive to position just slightly earlier than the cables spooled out, eliminating tangling. A plot of the encoder positions over time with the hybrid deploy method is shown in Figure 3.5. The play symbol on each encoder line shows when the motor started to drive to its position, and the diamond shows when the motor stopped and was applying maximum holding torque.

The hybrid deploy method was implemented as designed; however, the motors alone did not have enough holding torque to hold the arm cables in place during deployment. Each motor attempted to stop the cable at its commanded position, but by 800 ms all four motors had been dragged out and the arm was deployed straight (not the commanded configuration).

Consequently, we installed a 24V Electromagnetic Brake [46] on the back of each stepper motor to improve the holding torque. These brakes have a spring pressing two friction plates together, so when no power is supplied the motor is very difficult to turn. When 24V is applied,

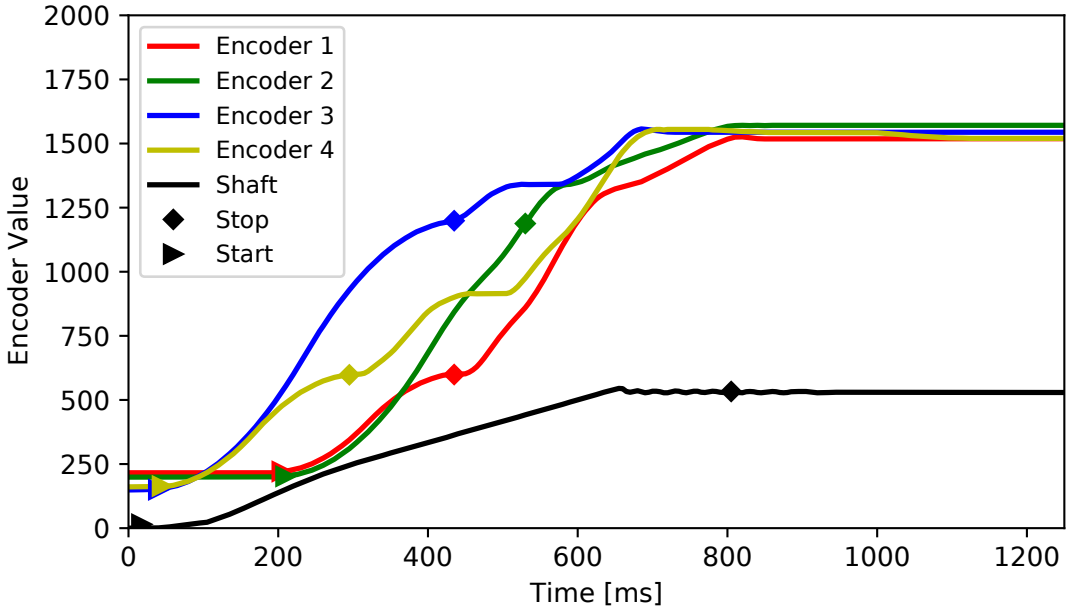


Figure 3.5: Encoder positions using hybrid deploy method

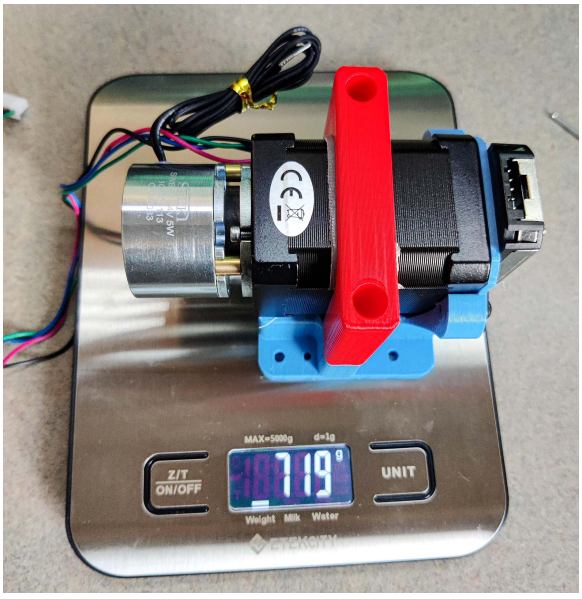


Figure 3.6: Improved motor assembly with dual-shaft stepper, electromagnetic brake, and stronger mount

the magnet pulls the friction plates apart and the motor spins freely as if the brake was not present. These brakes were designed to mount to dual-shaft stepper motors, so we also swapped out each stepper motor for the dual-shaft version and made the motor mounts stronger to support the added weight. The upgraded motor assembly without the cable spool or flange is shown in Figure 3.6 on a scale - each assembly without the flange or cable spool weighs 719 g (1.59 lbs).

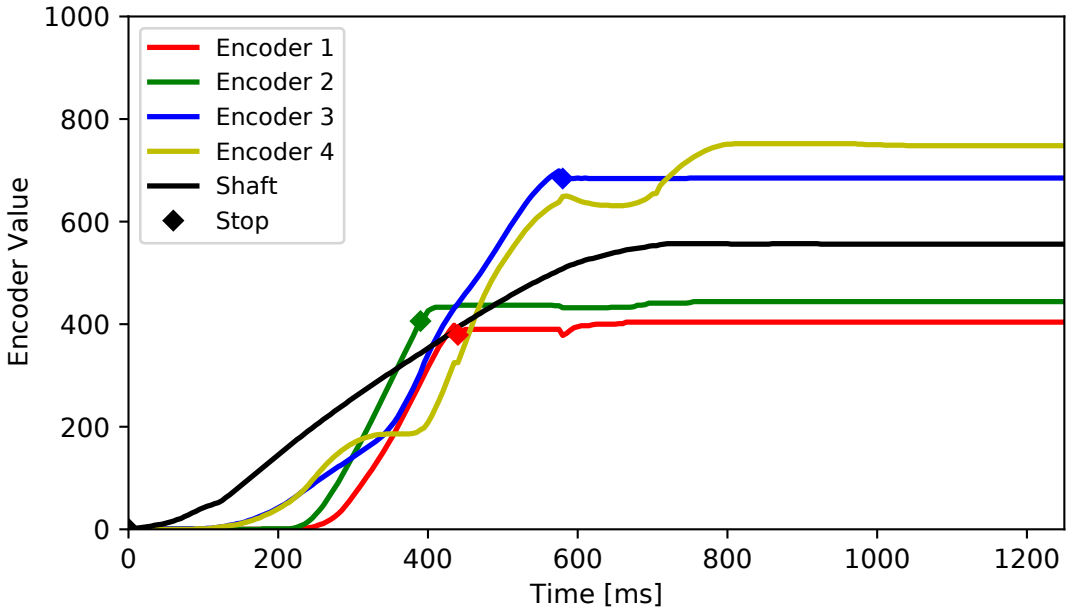


Figure 3.7: Encoder values after upgraded motor assembly

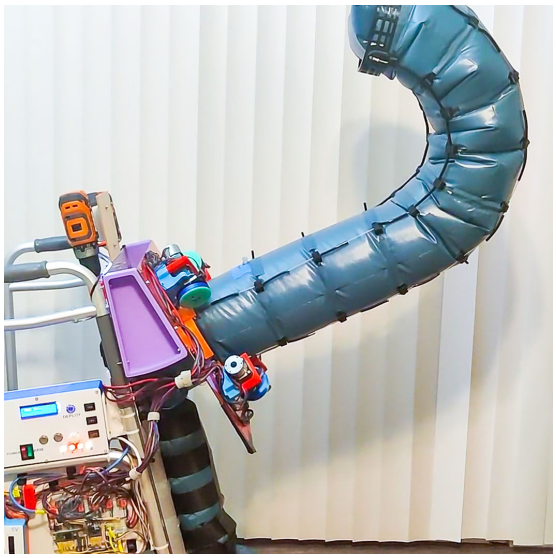
The motor performance results after the discussed changes were satisfactory - an example can be seen in Figure 3.7. The cable is pulled out to its desired position and then the brake is applied and the motor driver is enabled. The combination of these two subsystems working together was successfully able to stop the cable pulling during full-speed deploys.

3.1.3 Spacers to Improve Bending Performance

The first version of the arm did not have any spacers on the steel cable between the cable guides. We noticed that the cable guides often bunched up and the arm would not bend predictably or smoothly as a result. The guides would bunch up and cause the arm to make a sharp bend instead of the desired smooth bend. Therefore, we added a 2.5″ (6.35 cm) section of 4 mm spiral cable wrap [37] between each cable guide. There is 5″ (12.7 cm) between each cable guide, so the spacer was

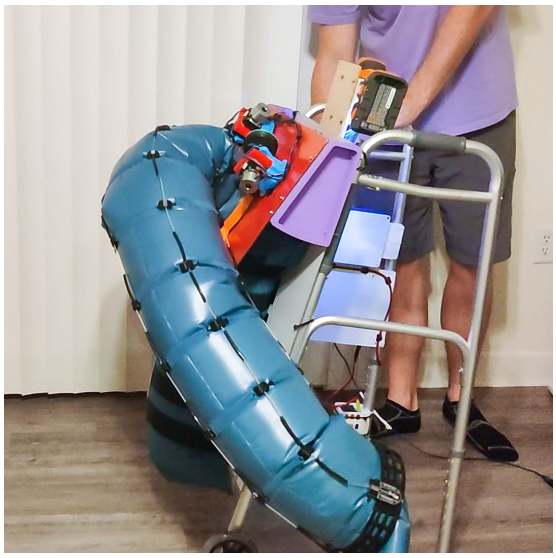
Figure 3.8: Spiral cable wrap spacers

cut to be half that distance. The cut spacers before being installed on the arm can be seen in Figure 3.8.

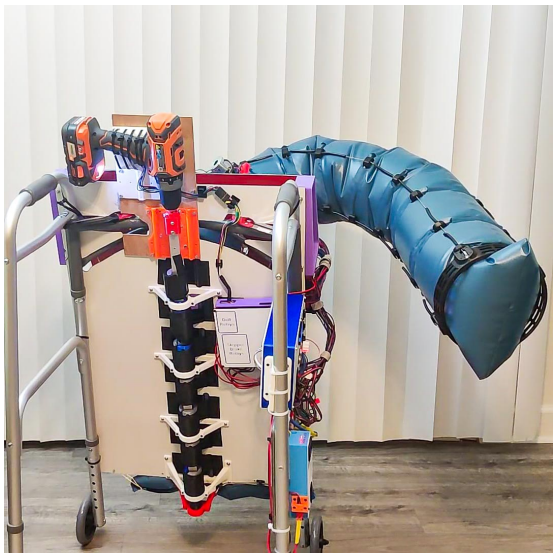


(a) Forward Up

(b) Forward Curl Up



(c) Left

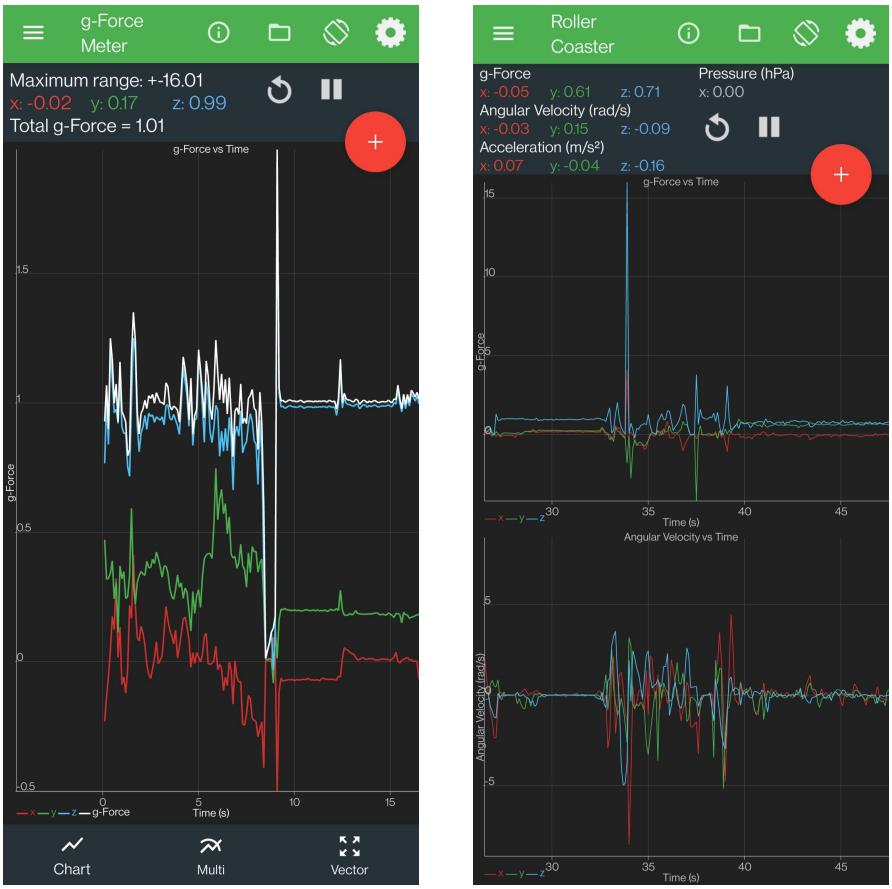


(d) Right

Figure 3.9: Selection of different configurations that the arm with spacers can achieve

3.1.4 Fall Data Collection Methods

We needed a reliable way to collect data on the severity of the fall with and without the airbag arm deploying in order to evaluate the effectiveness of the system. We made good use of an app called Physics Toolbox Sensor Suite [62] that uses internal smartphone sensors to collect, log, and export data files to CSV (comma separated values) files. This app can collect data from dozens of sensors inside a smartphone, but we were most interested in the g-force and linear acceleration data collection. The app also has some combination data collection modes where it can collect data from multiple sensors at the same time, such as Roller Coaster mode that logs g-force, linear acceleration, gyroscope, and barometer data all at the same time. Screenshots from the app in the g-force mode and roller coaster mode are shown in Figure 3.10.



(a) g-Force meter (b) Roller Coaster mode

Figure 3.10: Data collection modes in Physics Toolbox app

The phone we used with the data collection is a OnePlus 8, which can collect g-force data

up to ±16 g's at 400 Hz and linear acceleration at 200 Hz according to the app. We found that in combination modes it only logs the data to 2 decimal places, whereas in the singular data logging modes it logs to 4 decimal places. The 2 decimal places in the combination modes led to many repeated data points and inconsistent values for the forces at time of impact, so we chose to do the tests using single data collection modes.

We used a plastic mannequin (dummy) to simulate the mass distribution of a human user [63]. The dummy was 6′0" (183 cm) tall and weighed 12 lbs (5.4 kg). The phone was attached to the dummy using an old phone case glued to a 3D printed mount. The phone case allowed the phone to easily be attached and removed throughout testing, and the 3D printed mount securely held the phone case on the dummy.

The first mount (see Figure 3.11a) attached to the back of the dummy's neck using double sided tape and zip ties. The neck location gave clear data for the time of impact, but we determined that the lower back location would give more accurate data for the acceleration and g-forces on the dummy than the neck location. The lower back was closer to the center of mass for the dummy and better approximated the impact on the user's hip.

(a) Back of neck (b) Lower back

Figure 3.11: Phone mount locations for impact data collection

Several baseline fall tests were conducted with the dummy falling onto a futon cushion. These tests provided very consistent results as shown in Figure 3.12. For all of these baseline tests we attached a pair of ankle weights to the dummy's ankles totalling 8.2 lbs (3.7 kg) in order to minimize the rotation of the dummy as it fell. The tests were conducted without the arms attached to the dummy to improve consistency and to avoid damaging the shoulder socket and connection

piece from repeated falls. The initial impact is the tallest peak on the graph, and then the smaller peaks afterwards show the effect of the dummy bouncing on the cushion.

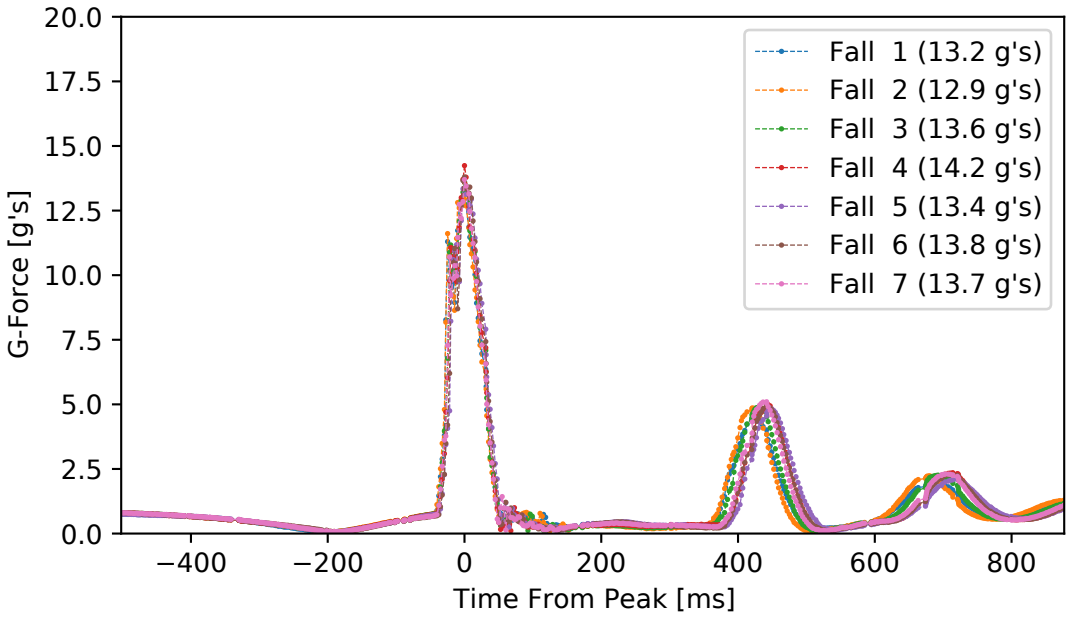


Figure 3.12: G-Force data from baseline tests of just the dummy with ankle weights falling onto the futon cushion

We also ran baseline tests with the dummy within the falling walker, but without the airbag deploying. The g-force or acceleration from these tests was much less consistent, and resulted in a larger impact or acceleration than the dummy falling alone. The data from these initial baseline tests with the walker are shown in Figure 3.13.

Detailed video analysis of the baseline fall with the walker showed that the walker fell faster than the dummy, resulting in more than a foot of separation between the dummy's chest and the walker frame before impact. This separation is shown in the video snapshot in Figure 3.14. The walker's fall was lessened by the futon cushion, but the dummy (with the data logging phone attached) hit the stationary walker and was not cushioned nearly as much. The dummy is a rigid body (plastic blow mold) and does not weigh a significant amount relative to the walker (12 lbs), so it did not push the walker down significantly on impact from the dummy's weight. The result was that our tests with the walker produced similar data to pushing the dummy over onto the hard floor instead of onto the futon cushion. This made it potentially difficult to compare the walker's performance when deploying the arm versus not deploying the arm.

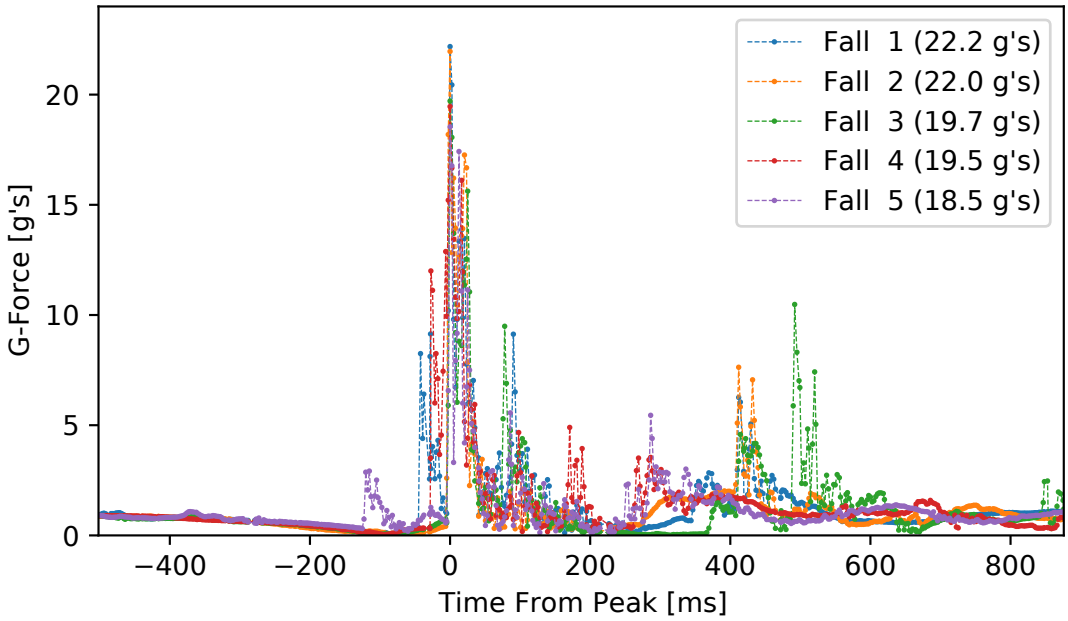


Figure 3.13: G-Force data from baseline tests with the walker and the dummy, showing worse results than without the walker

Our solution to this separation issue was to wrap a bungee around the dummy's upper legs and the walker frame. This bungee kept the dummy and walker together during the fall (much closer to what we would anticipate in a real human/walker fall), and resulted in much improved data.

At this stage, we were ready to begin testing of the overall system, deploying the airbag and measuring its performance. To summarize the changes we made while refining the system in early testing:

- Drill speed set to 1 (high torque, lower speed) and clutch at maximum
- Drill spun in the counterclockwise direction during deployment
- Added an electromagnetic brake to each motor
- Motors pulled to desired position and then brake engaged and driver enabled for holding torque
- Added spiral cable wrap spacers to the steel cables

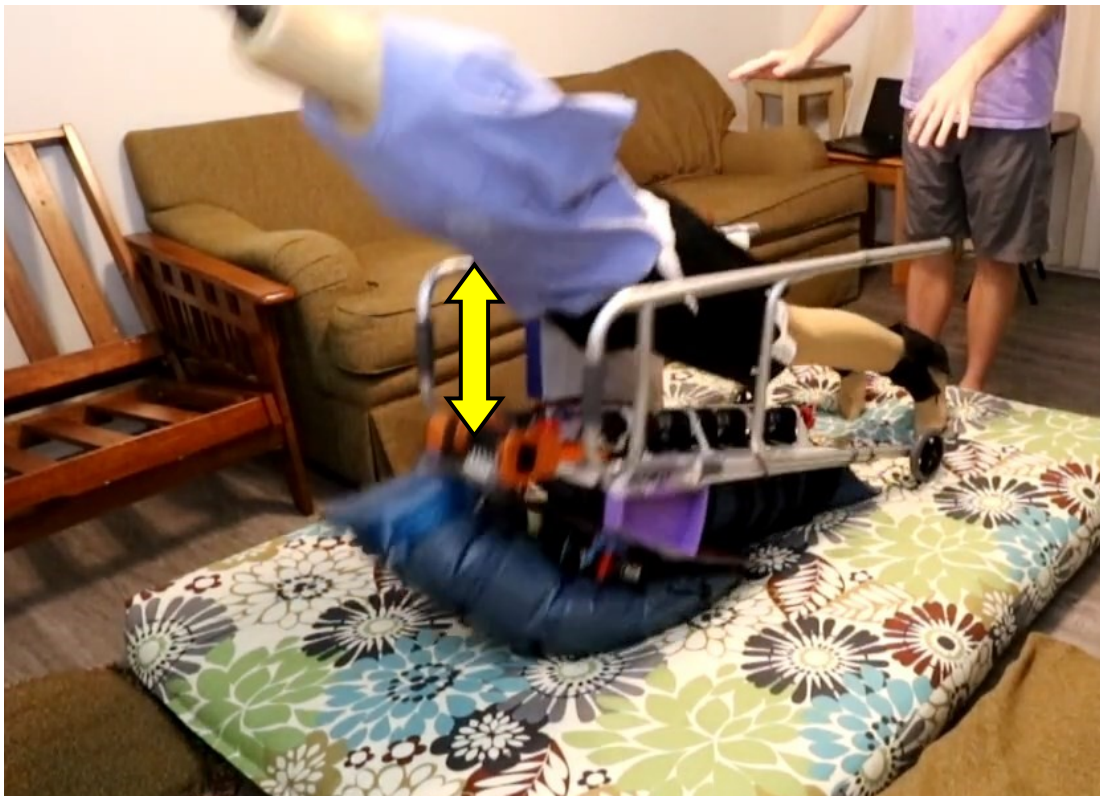


Figure 3.14: Separation at time of impact causing worse impact with the walker than without the walker

3.2 Final Deploy Tests

The improvements and fine-tuning described in the previous section culminated in a robust robotic prototype with consistent and quantitatively measurable results. In order to qualitatively and quantitatively evaluate the system, we conducted 5 trials in each direction of fall the airbag was designed to mitigate the effects of (forward, left, right). This number of trials was determined to be a reasonable number of trials based on other human-airbag experiments reported in the literature. For example, a wearable hip-airbag system for fall protection [64] was deployed 3 times but the group did not report any data on the effectiveness of the device. Similarly, the authors of [64] conducted 4 tests but only reported data on the inflation time and fall time, not the effectiveness of the device as an airbag. Other groups performed 1 test [15], no tests [65], [66], [67], or only simulated results [68]. Given there appears to be no standard number of trials or testing procedure for human-airbag fall protection system, we chose to perform 5 tests in each fall direction, and developed the testing procedure described below.

3.2.1 Testing Procedure

In each trial set, the walker was tipped forwards, left, and right. In each case the system was tested both with the airbag un-deployed (baseline test) and fully deployed. It was assumed throughout testing that real-time detection of the timing and direction and the falls was available. There is a fairly extensive literature on fall detection, e.g. [69], and fall detection was not the goal of this research. Therefore, the system was manually armed before the test and triggered when the IMU detected a tilt beyond a threshold. The arm configuration was chosen by the software based on the direction of fall imparted to the walker.

The dummy was attached to the walker using bungee cords to simulate the mass distribution of a human user. Attached to the dummy was the cell phone running the Physics Toolbox Sensor Suite measuring the g-force experienced by the phone. The robotic system logged data from the encoders and other key events (drill stopped, brakes engaged). Each test was recorded on video at 60 frames per second (FPS).

3.2.2 Baseline Tests

Baseline tests were conducted only for the left and forward directions. During the baseline tests, the dummy and walker were pushed over onto a futon cushion without the arm deploying. The g-forces measured during the left baseline tests can be seen in Figure 3.15, and a typical progression images from the fall can be seen in Figure 3.16. Note the consistency of the results across tests. The walker and dummy impacted the surface at approximately the same time (0 ms on the figure), bounced slightly (100 ms to 300 ms), then settled onto the surface.

Baseline tests in the right direction were presumed to be equivalent to baseline left tests. We chose to refrain from performing baseline tests in the right direction out of concern for the electronic components attached to the right side of the walker.

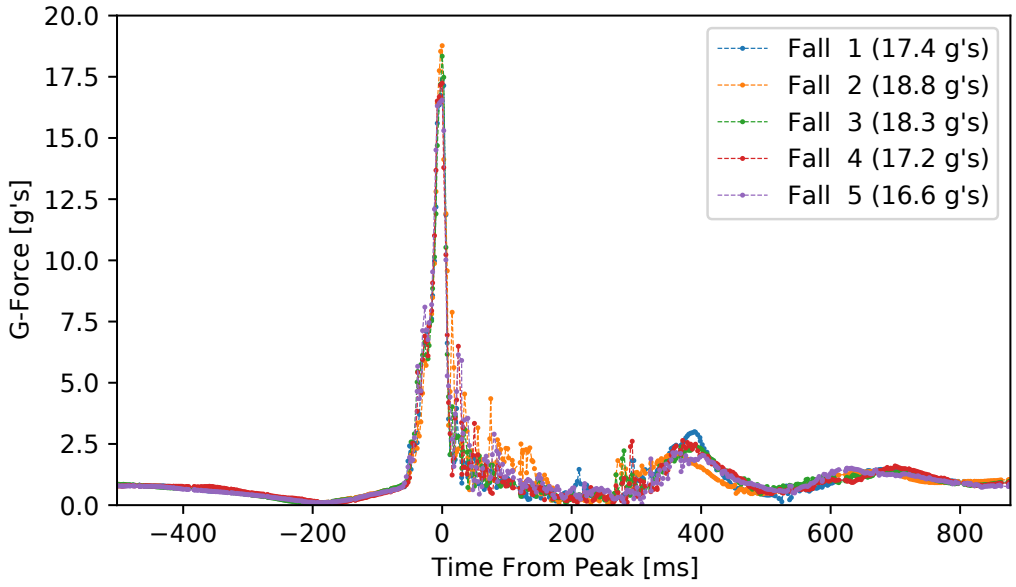


Figure 3.15: Data logging - g-forces experienced by the dummy during baseline left tests

Baseline tests in the forward direction were conducted in the same fashion. The results of the forward tests (Figure 3.17) were less consistent than the baseline left tests. Video analysis showed this was likely because the walker rolled forward and the dummy's head region impacted the ground in front of the cushion. The roll, impact, and any subsequent impacts were slightly more varied than the baseline left tests. Further, the peak impact for the baseline forward tests occurred when the dummy's head impacted the floor, instead of the initial impact of the walker on the cushion. A typical progression from video screenshots during a baseline forward fall is shown in

Figure 3.16: Progression of typical baseline left fall

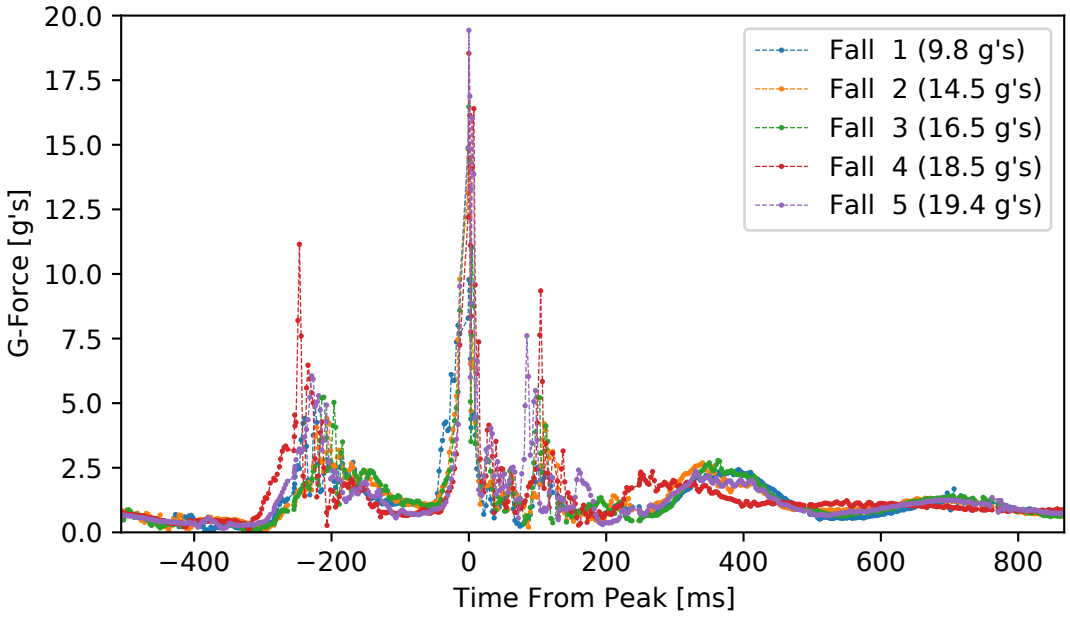


Figure 3.17: Baseline forward tests g-forces

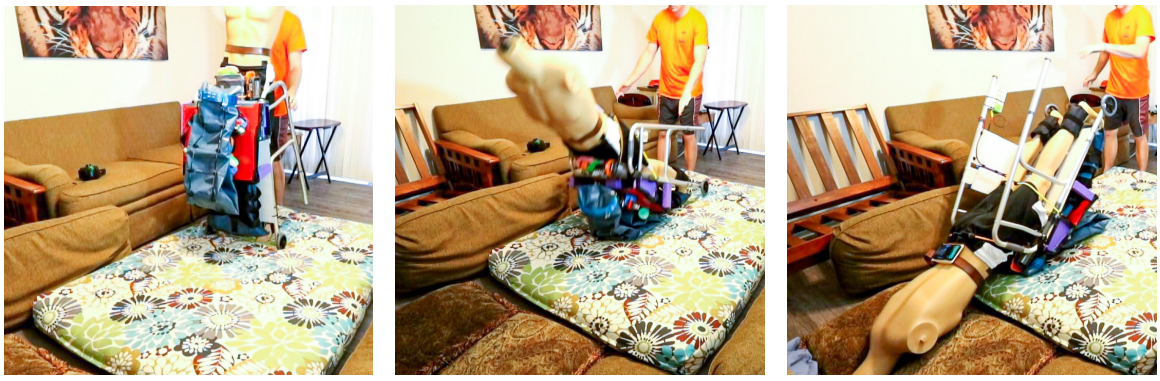


Figure 3.18: Progression of typical baseline forward fall

Figure 3.18. The peak g-force from forward fall 1 was significantly lower than the other tests, likely because the walker did not tip forward far enough for the dummy's head to hit the ground.

The baseline tests resulted in an average peak g-force of 17.7 g's in the left direction, and 15.7 g's in the forward direction. It is important to note that the actual peak g-force of most baseline tests was higher than our system could measure - the phone's sensors could only detect ±16.0 g's in each axis (x, y, z), and most baseline tests maxed out an axis at 16.0 g's. The peak g-force listed is the peak total g-force ($\sqrt{x^2 + y^2 + z^2}$) and for most tests only one axis changes significantly throughout the fall, but other axis of measurement have slight variations between the tests. This explains why there is slight variation between the baseline tests despite an axis being maxed out at ±16.0 g's. With that in mind, the relative performance of the system during deploys versus baseline is better than our data shows.

When the baseline tests were complete and showed consistent data, we moved on to conducting tests with the system deploying in the desired direction. The same test procedure was followed, but the system was armed before the test and deployed in real-time.

For the left-sided and right-sided fall tests, the arm was augmented with a small inflatable ring. This aided in stabilizing the dummy following the fall. Future versions of the walker could feature a redesigned arm that includes a larger end section that inflates with the rest of the system.

Preliminary tests showed similar performance when the arm was bent upward versus downward for the forward tests. All of the forward tests were completed with the arm deploying forward and bent upward.

3.2.3 Left Deploy

The results of the left deploy tests are shown in Figure 3.19. A progression of images during a typical left fall and deploy test are shown in Figure 3.20.

The left tests resulted in a maximum peak g-force of 8.9 g's, and a minimum peak g-force of 5.8 g's. The average peak g-force from the left fall tests was 7.2 g's.

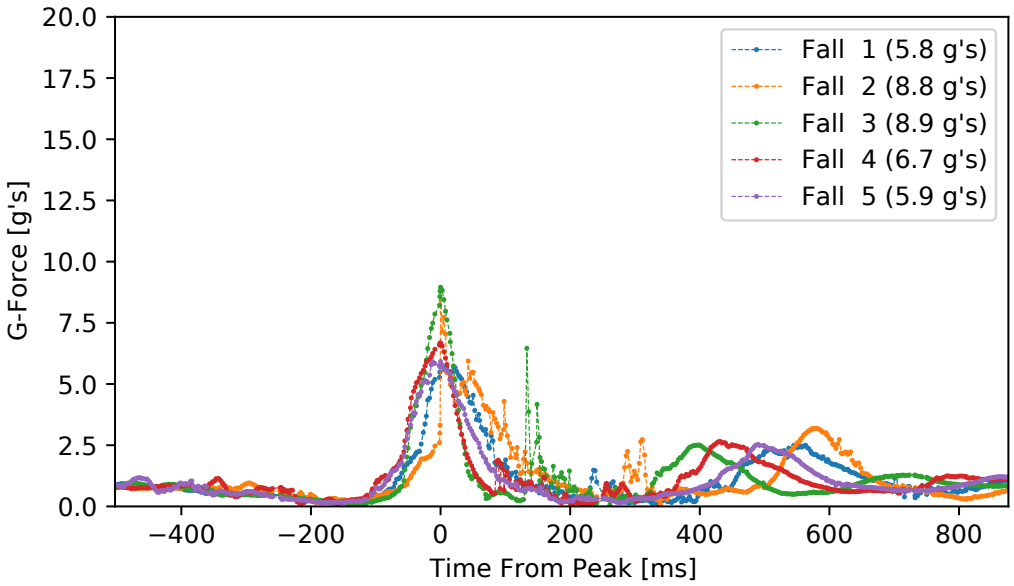


Figure 3.19: Left deploy g-forces over five tests

Figure 3.20: Progression of a left fall with the system deploying.

3.2.4 Right Deploy

The results of the right deploy tests are shown in Figure 3.21. A progression of images during a typical right fall and deploy test are shown in Figure 3.22.

The right tests resulted in a maximum peak g-force of 8.8 g's, and a minimum peak g-force of 6.6 g's. The average peak g-force from the left fall tests was 7.8 g's.

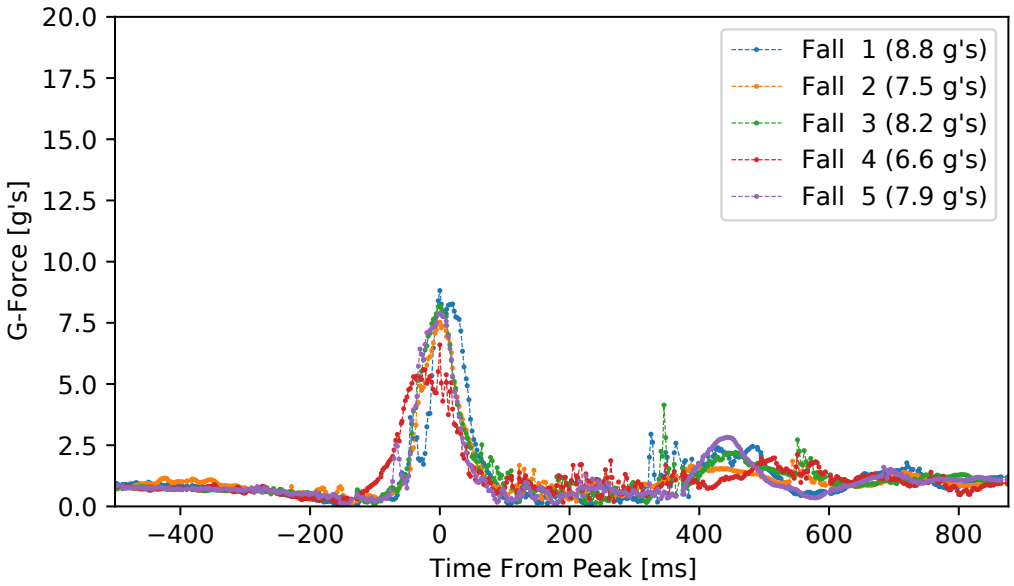


Figure 3.21: Right deploy g-forces over five tests

Figure 3.22: Progression of a right fall with the system deploying.

3.2.5 Forward Deploy

The results of the forward deploy tests are shown in Figure 3.23. A progression of images during a typical forward fall and deploy test are shown in Figure 3.24.

The forward tests resulted in a maximum peak g-force of 8.1 g's, and a minimum peak g-force of 6.0 g's. The average peak g-force from the forward fall tests was 7.3 g's.

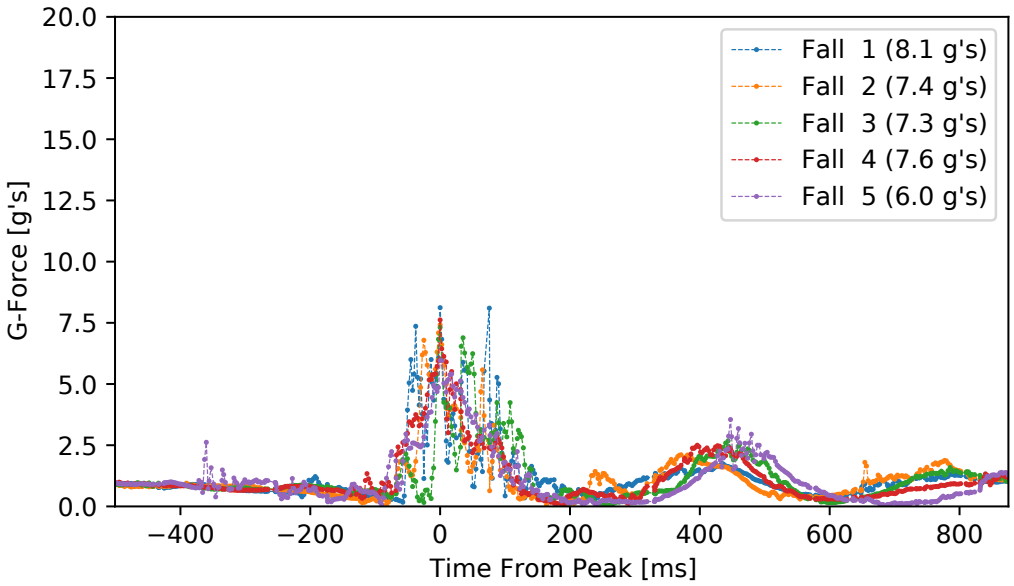


Figure 3.23: Forward deploy g-forces over five tests

Figure 3.24: Progression of a forward fall with the system deploying.

	Baseline Left	Baseline Forward	Deploy Left	Deploy Right	Deploy Forward
Test 1	17.4	9.8	5.8	8.8	8.1
Test 2	18.8	14.5	8.8	7.5	7.4
Test 3	18.3	16.5	8.9	8.2	7.3
Test 4	17.2	18.5	6.7	6.6	7.6
Test 5	16.6	19.4	5.9	7.9	6.0
Average	17.7	15.7	7.2	7.8	7.3

Table 3.1: Summary of experimental results - peak g-force measured during each fall

3.2.6 Summary of Results

Table 3.1 summarizes overall quantitative testing results, using data from the five representative tests of the system. On average, the measured maximum g-force of the deployed system in the forward falls is 46% of that for the baseline, i.e. undeployed, case. The average measured maximum g-force for the deployed system in left/right falls is 42% of that for the baseline case. The reduction in measured impact demonstrates the effectiveness of the airbag system.

3.3 Variable Weight Tests

After completing the set of tests with the dummy alone, we conducted tests with additional weight on the dummy to more closely simulate the mass of a human user. We added a backpack to each of the front and the back of the dummy and put weights inside of the backpacks. As with previous tests, the dummy also had 8 pounds of ankle weights attached to its ankles and was attached to the walker with bungees. We conducted these tests in the forward direction.

We ran two tests with 25 pounds (11.3 kg) total in the backpacks (Figure 3.25), then ran two tests with 50 pounds (22.7 kg) total in the backpacks (Figure 3.26). A progression of images during a forward fall with the weighted backpacks is shown in Figure 3.27.

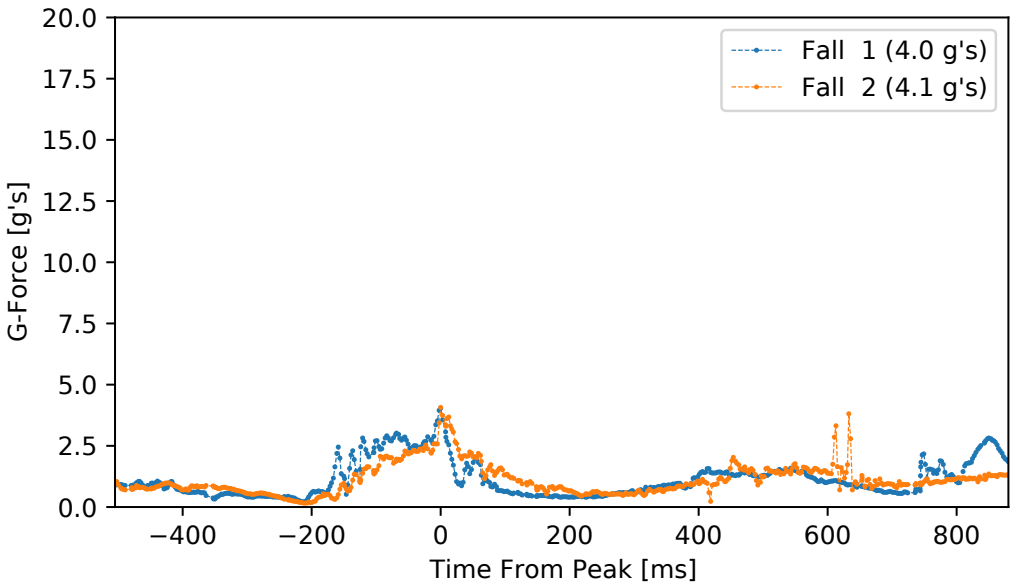


Figure 3.25: Weighted tests with 25 pounds added to the dummy

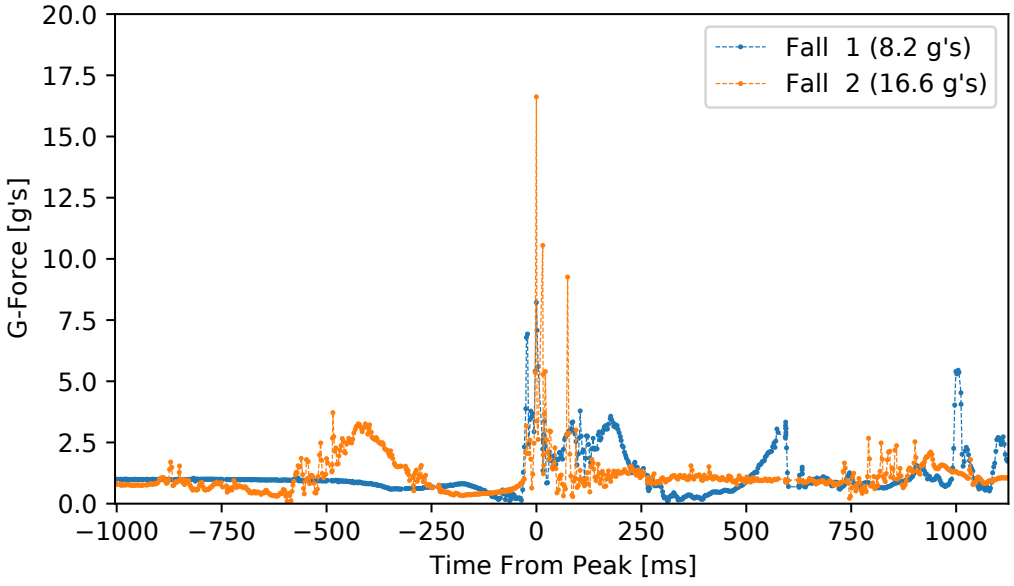


Figure 3.26: Weighted tests with 50 pounds added to the dummy

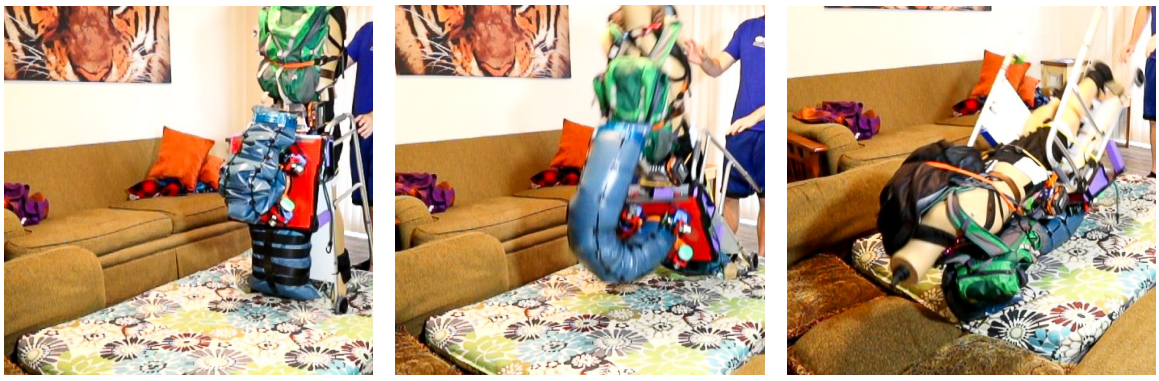


Figure 3.27: Progression of a forward fall with additional weight in the backpacks

The additional weight improved the system performance in the 25 pound tests. In the weighted tests, the dummy had sufficient mass and momentum to push back against the deployed arm, resulting in a smoother slowdown than in the earlier experiments, when the lightweight dummy was effectively "punched" by the arm and rapidly stopped.

The peak g-force from Test 1 of the 50 pound tests was similar to the unweighted test. Test 2 had a much lower initial peak impact of 3.7 g's as seen at time -450 ms in Figure 3.26, but the dummy and walker rolled forward and the dummy impacted the bare ground. This impact without any cushioning caused a peak of 16.6 g's as measured by the phone app. This rolling forward was partially due to the weight distribution on the dummy - all the weight was on the dummy's upper body, resulting in a top-heavy system that rolled forward.

Table 3.2 summarizes the weighted test results. On average, the measured maximum g-force of the deployed system in the 25 pound weighted forward falls is 26% of that for the baseline forward case. The average measured peak g-force of the initial impact was 5.9 g's which is 38% of that for the baseline forward case. The additional weight resulted in improved system performance.

	Weighted 25 pounds	**Weighted 50 pounds**
Test 1	4.0	8.2
Test 2	4.1	3.7

Table 3.2: Summary of variable weight results - peak g-force measured during the initial impact with the ground for each fall

Chapter 4

Conclusions and Discussion

In this thesis, a soft robotic airbag whose configuration can be controlled in real time to deploy in the direction of a forward or sideways fall was introduced. The core of the robotic airbag system was an air-filled, tendon-configured, continuum arm-like structure. The airbag was integrated into a conventional passive walker. A novel single motor system was used to deploy the airbag by transferring air from a reservoir into the arm in real time. Tests with a prototype demonstrated the effectiveness of the system in deploying through the desired range of configurations. The g-forces of impact were significantly and consistently reduced across the configuration space of the system. A paper based on this thesis was submitted to the International Conference on Soft Robotics 2022 [70].

The research represents the first airbag system that is robotic in the sense that the shape of a single deployed airbag can be selected based on the sensed direction of a fall. As such, it represents a novel contribution to the literature on robotic walkers. In creating and demonstrating the first continuum robotic airbag, the work is also a novel contribution to continuum robotics research. Overall, this thesis, in demonstrating the potential of a novel approach to protect the users of passive walkers from falls, represents a small but significant step towards the development of new robotic technology that can enable safer lives for the elderly and infirm.

4.1 Future Work

The key areas for improving the applicability of our system relate to weight and size. The final system with the walker weighs 36.2 lbs (16.9 kg). The original walker weighs 7.5 lbs (3.4 kg) according to the Amazon product page, meaning our system adds 28.7 lbs (13.5 kg) to the walker. Most users of mobility walkers would struggle to lift our system over any bump or obstacle. One place weight could be reduced is the cable braking mechanism (stepper motor and electromagnetic brake). Each motor assembly (brake, motor, encoder, 3D printed mount) weighs 1.6 lbs (0.73 kg), which for the four assemblies adds up to 6.4 lbs (2.9 kg) of the robotic system. A custom braking mechanism could greatly reduce this weight. The AluPoly sheets attached to the walker contributed a similar amount, weighing 5.5 lbs (2.5 kg) total. These pieces could be replaced with more lightweight material.

The size of the complete system is also not ideal - the air reservoir and other components take up a large amount of space on the front of the walker, and they are not visually appealing. Switching to a different deploy method such as CO_2 or HPA could improve this issue, resulting in a less conspicuous design. The arm could be folded up in a discrete pouch before deploying.

Additionally, alternative geometries of the continuum arm should be explored. The arm could be redesigned to include a large end piece to minimize roll after a fall, and/or to better wrap around the user for a left or right fall. Different shapes or additions to the arm could enhance the performance of the system. The number and arrangement of tendons configuring the airbag should be also be investigated. The method of manufacturing the fabric arm of the system could be improved upon to reduce variability in speed and tool-path when guiding the soldering iron by hand.

In additional to improvements in the components of the existing system, its the core functionality could be modified to act more as a helping hand than a vehicle airbag. The system could be modified so it deploys more slowly and can push off against walls or environmental obstacles. Potentially, the arm could even propel the walker around by sequentially inflating and deflating, pushing off the ground in front of it and driving the walker backwards. Walkers are used by many people across the globe, and ways in which a walker could be modified to improve the user's life have a huge potential market and impact.

Appendices

Appendix A Alternative Inflation Methods

This section describes the details and related calculations of some of our explorations of alternative methods to fully inflate the continuum arm from an uninflated initial state. The volume of air in the air bladder of our ultimate prototype is not considered because an air bladder is not needed if you use a different deploy method such as compressed air.

The arm in our system is 135 cm long, and measures 63 cm in circumference when fully inflated. The radius of the arm is calculated with:

$$r = \frac{63\,\text{cm}}{2 * \pi} = 10.02\,\text{cm} \approx 10\,\text{cm} \tag{1}$$

The cross-sectional area (A) of the arm is

$$A = \pi * r^2 = \pi * 10^2 = 314.2\,\text{cm}^2 \tag{2}$$

Using a radius of 10 cm, the volume (V) of air in the inflated arm is calculated by:

$$V = 135\,\text{cm} * 314.2\,\text{cm}^2 = 42417\,\text{cm}^3 \tag{3}$$

This volume is equivalent to $0.0424\,\text{m}^3$ or $2588\,\text{in}^3$ or $1.50\,\text{ft}^3$. For visualization, one cubic foot is approximately equal to 7.5 gallons, so to fill up our arm with water would take 11.25 gallons of water.

A.1 Compressed CO_2

A common method to inflate human airbags in research is by puncturing a compressed carbon dioxide (CO_2) cartridge as discussed in the Background section. These canisters contain highly compressed CO_2 in liquid form. As the liquid CO_2 is released it turns into a gas, and the energy required for this phase change makes the metal CO_2 cartridge very cold [71]. These cartridges are sold based on the weight of liquid CO_2 in grams - a 12-gram cartridge has 12 g of liquid CO_2 inside of it. We can use the ideal gas law to calculate the volume of the gas once it expands. For these calculations we will neglect any effects on the change of temperature on the volume of the gas.

The ideal gas law is

$$p * V = n * R * T \tag{4}$$

In the above equation, p is the pressure of the system. V is the volume. n is the number of moles. R is the ideal gas constant. T is the temperature of the system.

Rearranging to solve for volume:

$$V = \frac{1 * n * R * T}{p} \tag{5}$$

We can use the above equation to calculate the resultant gas volume (V) in liters from 1 gram of pressurized liquid CO_2.

The molar mass of CO_2 is 44.01 g/mol [72]. For x grams of CO_2, the number of moles (n) is $x/44.01$. In the following, we use a room temperature (T) of $72°F \approx 295K$ and a standard pressure (p) of 1 atm. R is the ideal gas constant, $0.0821L * atm/(mol * K)$. Plugging these values into the equation:

$$V = \frac{0.0821\,\text{L*atm/(mol*K)} * 295\,K}{44.01\,\text{g/mol} * 1\,\text{atm}} * \frac{1\,\text{cm}^3}{0.001\,\text{L}} = 550.3\,\text{cm}^3/\text{g} \tag{6}$$

To calculate the grams of CO_2 to fill the 42417 cubic centimeters of our arm:

$$x\text{g} = \frac{42417\text{cm}^3}{550.3\,\text{cm}^3/\text{g}} = 77.08\,\text{g CO}_2 \tag{7}$$

A common size of CO_2 cartridges near this size is 88g-90g cartridges, implying we would only need one cartridge per deploy. The 88-gram cartridges are approximately $1''$ diameter by $6''$ long and cost around \$8 per cartridge [73] [74] . A size comparison of different compressed gas cartridges can be seen in Figure A.1.

You could also obtain the required 77 grams of CO_2 by triggering three 25-gram cartridges ($\approx$ \$4.40/cartridge, \$13.20 per deploy), five 16-gram cartridges ($\approx$\$1.70/cartridge, \$8.50 per deploy) or seven 12-gram cartridges ($\approx$\$0.95/cartridge, \$6.65 per deploy). Each cartridge would need its own puncture mechanism and air tubes routed to the arm. We conducted around 80 full deploy tests with the prototype in this thesis, which would have cost \$640 in cartridges at \$8 per deploy. The cost of the cartridge would possibly not be an issue for a commercial product deploying once,

Figure A.1: Size comparison of compressed gas cartridges

but was an important factor to consider here for research costs and feasibility.

Refillable CO_2 tanks are commonplace in paintball, as small as 12 oz (340 g) [75]. It was not clear however whether the tanks could release the required volume of air for our continuum arm all at once. It was possible that the tank's pin valve might freeze up before it can release a quarter of the tank. The release system would also need to have a way to stop releasing CO_2 after the arm is fully inflated, or the arm would need to be designed with a pressure release valve.

A.2 High Pressure Air

A form of compressed air called High Pressure Air (HPA) is a common alternative to CO_2 used in paintball. The HPA tanks are filled with air at the tank's rated pressure from 3000 to 4500 pounds per square inch (psi), then is released out of a regulator on the tank typically at 850 psi [76]. HPA tanks are categorized by their Cubic Inches (ci) and their max PSI rating. A common tank size and pressure combination is 13/3000 (13 in^3 when compressed at 3000 psi).

To estimate the uncompressed volume of air comparable to deploying, we use Boyle's Law which states that the volume of a gas changes proportionally to the pressure of the gas:

$$p_a V_a = p_c V_c \tag{8}$$

Setting p_c (pressure compressed) to 3000 psi and volume compressed (V_c) to 13 cubic inches, we use 1 atmosphere as the pressure uncompressed (p_a) which is 14.7 psi. Substituting these values

and rearranging terms we obtain

$$p_a = \frac{3000\,\text{psi} * 13\,\text{in}^3}{14.7\,\text{psi}} = 2653\,\text{in}^3 \tag{9}$$

This is slightly more than the volume of air calculated in (3) above. This shows that if the entire HPA tank could be released during the deploy of the arm, it would be sufficient to fully inflate the arm. There are also larger tanks sold up to 48 cubic inches when compressed, so the volume of air can be provided by selecting an appropriate tank or tanks. (This calculation is not exact, because it does not take into account moisture content of the compressed air.)

Practically, there are several potentially problematic issues to using high pressure air. The 850 psi regulator would limit the release pressure to 850 psi instead of the full 3000 psi, likely keeping the tank from releasing the full volume of air within the 500 ms needed for deployment as the user falls. This could potentially be compensated for by putting several HPA tanks in parallel and releasing them at the same time. The tanks are not designed to release more than a puff of air at one time however, so significant care would need to be taken when prototyping with compressed air tanks to make sure the tanks did not explode.

We chose not to use HPA (or CO_2) for our project due to the physical danger to the researchers and the potential damage to the continuum arm if too much air was accidentally released at one time. A design using HPA would need to implement pressure valves and other safety measures for over-pressure events.

A.3 Vehicle Airbags

Airbags in vehicles lower the impact of a crash by stretching out over a longer period of time and by spreading the physical impact of a larger area of the body. Vehicle airbags deploy within 35 ms of an accident. Vehicle airbag systems consist of an accelerometer, a heating element, an ignition compound, the main propellant, and the airbag itself. In the event of an accident, the heating element rapidly heats up and ignites the ignition compound. The heat from the ignition compound burning starts the decomposition of the main propellant - sodium azide (NaN_3). The chemical equation for the reaction when sodium azide is heated is:

$$2\ NaN_3 \longrightarrow 2\ Na + 3\ N_2 \tag{10}$$

This reaction produces a large volume of nitrogen gas from a small amount of solid sodium azide. A standard front passenger airbag has an inflated volume of around 140 L (140000cm^3), which is approximately 3.3 times larger than our continuum arm [77]. A variety of other chemicals are mixed in with the sodium azide to form nontoxic compounds from the sodium (Na) [78].

To compare this inflation method to the other methods we can calculate the amount of solid sodium azide needed to fully inflate a standard airbag. NaN_3 has a molar mass of 65.0 g/mol, and the standard molar volume of NaN_3 is 22.4 L/mol at STP. Thus, the amount of nitrogen gas needed to fully inflate the standard airbag is:

$$\frac{140\,\mathrm{L}}{22.4\,\mathrm{L/mol}} = 6.25\ \text{mol}\ N_2 \tag{11}$$

Substituting 6.25 moles into equation 10, we can calculate the mass of solid sodium azide as follows:

$$6.25\ N_2 * \frac{2\ NaN_3}{3\ N_2} * \frac{65.0\,\mathrm{g}}{NaN_3} = 270.8\,\mathrm{g}\ NaN_3 \tag{12}$$

It therefore would take 270.8 g of solid sodium azide to inflate a standard 140 L car airbag. With a density of 1.85 g/cm^3, 271 g would take up 147 cm^3 (0.147 L). This volume is significantly less than the volume of HPA or CO_2 to inflate a similar volume. Airbag systems come in a variety of shapes and sizes to fit different areas of the vehicle (frontal airbags, side torso airbags, curtain airbags, knee airbags) and an appropriate size could be chosen for a given project.

Each replacement airbag costs $200 to $700 not including labor [79]. We deployed our project dozens of times while refining our system. Buying a new airbag for every deployment would have been cost-prohibitive during development.

Even though vehicle airbags could feasibly provide the needed inflation volume, we felt they were too dangerous to be safely experimented with. Airbags are not designed to be used outside of a vehicle, and there are very few resources available online relating to using airbags in any other application. The manufacturing and sale of commercial airbags is highly regulated, and typically they are only sold for installation in a specific vehicle. The reaction and deployment of vehicle airbags is explosive in nature and would be dangerous to experiment without appropriate training and safeguards in place. The speed at which they deploy could have caused issues for the continuum aspect of this project, since the hardware would need to be able to stop a cable deploying at a very

rapid speed if it was inflated with a vehicle airbag.

Appendix B Software Details

Select code functions are listed below. The full Arduino code used for this project can be found at `https://github.com/jakabo27/Walker-Continuum-Robot-Airbag`

Along with the main program with all systems integrated, several "Tester" programs are listed in the library. These programs test an isolated part of the system, such as printing the value of the buttons and switches to help determine which inputs need to be inverted or which switches aren't wired correctly.

B.1 Deploy

The function **DeployingNow()** deploys the continuum arm.

```
void DeployingNOW()
{
  // Trigger drill relays to start the drill accelerating
  DrillSetSpeed(deployDrillSpeed);
  DrillSwitchON();
  DrillCCW();

  // Read shaft encoder and reset it to 0.
  //   (Other encoders set in ArmedAndReady.ino with homing
     functionality)
  shaftEncoder.readAndReset();

  // Release all stepper motors to spin freely
  AllStepperMotorsOFF();
  BrakesAllOFF();

  // LED strips to red
  setAll(RED);  leds.show(); digitalWrite(DEPLOY_LED, LOW);

  // Update LCD
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Deploying ");
  lcd.setCursor(0,1);
  lcd.print(fallDirNames[fallDirection]);
  LCDUpdate = 0; //Reset Timer

  // Timers intialize
  elapsedMillis sinceDeployStart;
  elapsedMillis printTimer; //For printing to serial monitor
  elapsedMicros logTimer; //For data logging
  elapsedMillis IMUTimer;

  // Zero the data logging array
  memset(logData, 0, sizeof(logData));
  logCounter = 0;

  // Determine the ENCODER end value for each stepper motor.
  int E1endValue = goalEncoderSpots[fallDirection-1][0];
  int E2endValue = goalEncoderSpots[fallDirection-1][1];
  int E3endValue = goalEncoderSpots[fallDirection-1][2];
  int E4endValue = goalEncoderSpots[fallDirection-1][3];

  // Stepper to encoder conversion for driving motors to their spot
  //         Account for movement after "home" spot
  int S1endValue = (E1endValue - encoderS1.read()) * 200/EncoderCPR;
  int S2endValue = (E2endValue - encoderS2.read()) * 200/EncoderCPR;
```

```
47  int S3endValue = (E3endValue - encoderS3.read()) * 200/EncoderCPR;
48  int S4endValue = (E4endValue - encoderS4.read()) * 200/EncoderCPR;
49
50  // Spot to drive each motor after
51  int E1driveValue = encoderS1.read() + deployWhenEncoderMoves;
52  int E2driveValue = encoderS2.read() + deployWhenEncoderMoves;
53  int E3driveValue = encoderS3.read() + deployWhenEncoderMoves;
54  int E4driveValue = encoderS4.read() + deployWhenEncoderMoves;
55
56  //Variables to keep track of if a stepper is driving or not
57  bool S1driving = 0; bool S2driving = 0;
58  bool S3driving = 0; bool S4driving = 0;
59  bool S1Braked = 0; bool S2Braked = 0;
60  bool S3Braked = 0; bool S4Braked = 0;
61
62  // Deploy logic
63  bool stillDeploying = 1;
64  bool drillTriggerReleased = 0;
65  bool drillUnwinding = 0;
66  bool drillCompletelyDone = 0;
67
68  //Variables to keep the encoder values in
69  int E1val = 0;  int E2val = 0;
70  int E3val = 0;  int E4val = 0;
71  int EShaftval = 0;
72
73  // Reset deploy timer to zero
74  sinceDeployStart = 0;
75
76  while(stillDeploying)
77  {
78
79    //Read encoders
80    E1val = encoderS1.read(); E2val = encoderS2.read();
81    E3val = encoderS3.read(); E4val = encoderS4.read();
82    EShaftval = shaftEncoder.read();
83
84    //Turn on and lock each stepper as it spools out to the goal
         location
85    if(E1val >= E1endValue) {digitalWrite(S1_ENABLE, 0);  S1driving = 0;
          if(!S1Braked){BrakeON(1); S1Braked = 1;}} else {};
86    if(E2val >= E2endValue) {digitalWrite(S2_ENABLE, 0);  S2driving = 0;
          if(!S2Braked){BrakeON(2); S2Braked = 1;}} else {};
87    if(E3val >= E3endValue) {digitalWrite(S3_ENABLE, 0);  S3driving = 0;
          if(!S3Braked){BrakeON(3); S3Braked = 1;}} else {};
88    if(E4val >= E4endValue) {digitalWrite(S4_ENABLE, 0);  S4driving = 0;
          if(!S4Braked){BrakeON(4); S4Braked = 1;}} else {};
89
90    //Stop spinning the drill if it's spun X rotations
91    if(EShaftval <= shaftStepsBeforeStopping && !drillTriggerReleased)
92    {
```

```
93        DrillSetSpeed(0);
94        DrillSwitchOFF();
95        drillTriggerReleased = 1; // Only run this block once.
96      }
97
98      //Stop the drill after 500ms
99      if(sinceDeployStart > drillMaxDeployTime && !drillTriggerReleased)
100     {
101       DrillSetSpeed(0);
102       DrillSwitchOFF();
103       aw.digitalWrite(DRILL_DIR_PIN, 1);
104       drillCurrentDir = 1;
105       drillTriggerReleased = 1; // Only run this block once.
106      }
107
108     //Log data
109     if(logTimer >= logPeriod)
110     {
111       logTimer = 0; //Reset timer
112
113       logData[logCounter][0] = sinceDeployStart;
114
115       // Motor Encoders
116       logData[logCounter][1] = E1val;   logData[logCounter][2] = E2val;
117       logData[logCounter][3] = E3val;   logData[logCounter][4] = E4val;
118
119       // Drill
120       logData[logCounter][5] = EShaftval;
121       logData[logCounter][6] = drillTriggerReleased;
122
123       // Cable Stopped
124       logData[logCounter][7] = S1Braked;
125       logData[logCounter][8] = S2Braked;
126       logData[logCounter][9] = S3Braked;
127       logData[logCounter][10] = S4Braked;
128       logData[logCounter][11] = S1driving;
129       logData[logCounter][12] = S2driving;
130       logData[logCounter][13] = S3driving;
131       logData[logCounter][14] = S4driving;
132
133       // IMU data
134       logData[logCounter][15] = IMU_LR_RAW; // Tilt L/R
135       logData[logCounter][16] = IMU_FB_RAW; // Tilt F/B
136       logData[logCounter][17] = Axyz[2]; // Acceleration L/R
137       logData[logCounter][18] = Axyz[0]; // Acceleration F/B
138       logData[logCounter][19] = Axyz[1]; // Acceleration Z(?)
139
140       logCounter++;
141     }
142
143     //Update IMU data every few ms
```

```
144     if(IMUTimer >= 2 )
145     {
146       if ( imu.dataReady() )
147       {
148         refreshIMUData();
149         IMUTimer = 0;
150       }
151     }
152 
153     //Stop deploying after X seconds
154     if (sinceDeployStart > 1400)
155       stillDeploying = false;
156   }
157 
158   Serial.println("\n######␣DONE␣DEPLOYING␣######\n");
159 
160   //LEDs to green
161   setAll(GREEN);
162   leds.show();
163 
164   //Release all the cable spools
165   AllStepperMotorsOFF();
166   BrakesAllOFF();
167 
168   //Unwind the drill a bit
169   if(!drillTriggerReleased)
170   { // Can't change direction while trigger pulled in
171     DrillSetSpeed(0);
172     DrillSwitchOFF();
173     delay(30);
174   }
175   DrillClockwise();
176   DrillSwitchON();
177   DrillSetSpeed(20);
178   delay(500);
179   DrillSwitchOFF();
180   DrillSetSpeed(0);
181 
182   //Lastly, say we already deployed and need to be reset
183   alreadyDeployed = 1;
184   state = DEPLOYED_WAIT_FOR_RESET;
185   newStateFirstTime = 1;  oldState = state;
186 
187   PrintLog();
188 
189   DeployedWaitForReset();
190 
191 }
```

B.2 Check Inputs

The **CheckInputs()** function reads the input from the buttons, switches, and the IMU and stores them in global variables. A rolling average is applied to the IMU data to help smooth out outliers and voltage spikes.

```
void CheckInputs()
{
  // Dual Switch A (Middle Switch)
  if(!digitalRead(DUAL_A_2_PIN))
    dualA_State = 1;
  else if (!digitalRead(DUAL_A_1_PIN))
    dualA_State = 2;
  else
    dualA_State = 0;

  // Dual Switch B (Bottom Switch)
  // Wired so either position (1 or 2) is ON, middle is OFF
  if(!digitalRead(DUAL_B_12_PIN))
    dualB_State = 1;
  else
    dualB_State = 0;

  // Dual Switch C (Top Switch)
  // Wired so either position (1 or 2) is ON, middle is OFF
  if(!digitalRead(DUAL_C_12_PIN))
    dualC_State = 1;
  else
    dualC_State = 0;

  // ArmSwitch
  armSw_State = !digitalRead(ARMSW_PIN);
  if(armSw_State == 0)
    alreadyDeployed = 0; //Only fully reset after armSw goes back to 0.

  // Power Switch
  powSw_State = !digitalRead(POWSW_PIN);

  // Deploy button
  deployButton.update();
  deployButton_State = deployButton.isPressed();

  // Rotory Knob Button
  knobButton.update();
  knobButton_State = knobButton.isPressed();

  //IMU
  if ( imu.dataReady() )
  {
    imu.getAGMT();
```

```
45      get_scaled_IMU(Axyz, Mxyz); // In HelperFunctions.ino
46      Mxyz[1] = -Mxyz[1]; // Align magnetometer with accelerometer
47      Mxyz[2] = -Mxyz[2]; //  (reflect Y and Z)
48
49      //Save the raw non-averaged data
50      IMU_LR_RAW = Mxyz[2];
51      IMU_FB_RAW = Mxyz[0];
52
53      //Filter the IMU data with a rolling average
54      // Subtract the last reading
55      IMU_LR_total = IMU_LR_total - IMU_LR_readings[readIndex];
56      IMU_FB_total = IMU_FB_total - IMU_FB_readings[readIndex];
57
58      // Read from the sensor
59      IMU_LR_readings[readIndex] = Mxyz[2];
60      IMU_FB_readings[readIndex] = Mxyz[0];
61
62      // Add the reading to the total
63      IMU_LR_total = IMU_LR_total + IMU_LR_readings[readIndex];
64      IMU_FB_total = IMU_FB_total + IMU_FB_readings[readIndex];
65
66      // Advance to the next position in the array
67      readIndex = readIndex + 1;
68      // Calculate the average
69      if (readIndex >= numReadings) { readIndex = 0; }
70      IMU_LR_average = IMU_LR_total / numReadings;
71      IMU_FB_average = IMU_FB_total / numReadings;
72
73      // Update global variables
74      IMU_LeftRight = IMU_LR_average;
75      IMU_ForwardBack = IMU_FB_average;
76    }
77  }
```

B.3 Determine Fall Direction

The system determines which direction it is falling by comparing the IMU readings to threshold values. The system saves the most recent IMU average at the time the system is armed as the "Stable" value, and compares all subsequent readings to that value.

The global variables related to the IMU thresholding are initialized before the **setup()** function:

```
// Globals for most recent IMU reading
float IMU_LeftRight = 0;
float IMU_ForwardBack = 0;

//Used for saving the "Stable" value at the time the system is armed
float IMU_Stable_LR = 0;
float IMU_Stable_FB = 0;

//How much change in that direction to count as falling.
float FallLeftDelta= 0.08;
float FallRightDelta = 0.11;
float FallForwardDelta = 0.10;
```

The **determineFallDirection()** function then compares the most recent IMU average to the stable value to determine if the system is falling in a particular direction.

```
//Figure out which way the walker is falling (or not)
int determineFallDirection()
{
  fallDirection = STABLE;

  if(IMU_LeftRight < (IMU_Stable_LR - FallRightDelta))
  {
    fallDirection = FALL_RIGHT;
  }
  else if(IMU_LeftRight > (IMU_Stable_LR + FallLeftDelta))
  {
    fallDirection = FALL_LEFT;
  }
  else if(IMU_ForwardBack < (IMU_Stable_FB - FallForwardDelta))
  {
    fallDirection = FALL_FORWARD;
  }
}
```

B.4 Determine System State

The **DetermineState()** function determines the overall system state (such as Armed, Standby, MotorTuning, etc.) based on the status of the system (previously deployed or not) and the state of the switches. The state determination is purposefully designed so that some states will be selected before others. For instance, the system will go into the MotorTune state even if the ARM switch is toggled on and the walker is tilting sideways. This setup helps prevent accidental deploys of the system before it is desired.

```
void DetermineState()
{
  // Update global variables
  CheckInputs();
  determineFallDirection();

  // Motor tuning functions
  if (dualA_State == 1)
    state = TUNE_ONE_STEPPER;
  else if (dualA_State == 2)
    state = TUNE_ALL_STEPPERS;
  else if (dualB_State != 0)
  state = TUNE_DRILL_MOTOR;

  // Require resetting the ARM switch upon boot
  //     (alreadyDeployed is set to 1 in the setup file)
  else if (alreadyDeployed == 1 && armSw_State == 1)
    state = DEPLOYED_WAIT_FOR_RESET;

  else if (armSw_State == 1 && dualA_State == 0 &&
           dualB_State == 0 && alreadyDeployed == 0)
    state = ARMED_AND_READY; //Listening to IMU sensor for falling.

  else if (armSw_State == 0 && dualA_State == 0 && dualB_State == 0)
    state = STANDBY;
  else
    state = UNDEFINED;

  //Keep track of if it's the first time in the Arm state or not
  if(state != ARMED_AND_READY)
    armedReadyFirstTime = 1;

  if(state != oldState)
  {Serial.println("New State:  " + String(state));
    newStateFirstTime = 1;   }

  oldState = state;
}
```

B.5 Loop

The **loop()** function uses the previously determined state to run the correct function for the system. In Arduino programs, the loop() function is required and loops infinitely as long as the microcontroller is powered.

```
void loop()
{
  DetermineState();

  switch (state) {
    case STANDBY:
      Standby(); //keep alive function basically
      break;
    case TUNE_ALL_STEPPERS:
      TuneAllSteppers();
      break;
    case TUNE_ONE_STEPPER:
      TuneOneStepper();
      break;
    case TUNE_DRILL_MOTOR:
      TuneDrillMotor();
      break;
    case ARMED_AND_READY:
      ArmedAndReady();
      break;
    case DEPLOYING:
      DeployingNOW(); //blocking function
      break;
    case DEPLOYED_WAIT_FOR_RESET:
      DeployedWaitForReset();
      break;
    case UNDEFINED:
      //continue to check the switches until a valid state exists
      break;
    default:
      //same as UNDEFINED.
      break;
  }
}
```

B.6 Drill

Power drills are programmed to turn off after a certain period of inactivity to conserve battery. For our particular drill, it turned off after 10 seconds of inactivity. To circumvent this feature, our system moved the drill every 8 seconds. The speed was set very slow (4/100) so it was not noticeable that the drill moved, and the direction was inverted after every toggle so that the drill would not creep and slowly wrap up the cables. The **KeepDrillAwake()** function below implements this functionality.

```
void KeepDrillAwake()
{
  DrillSwitchON();
  SetDrillSpeed(4);
  delay(40);
  SetDrillSpeed(0);
  delay(50);

  //Flip direction (unless armed and ready to deploy CCW)
  if(!armSw_State)
  {
    DrillFlipDirection();
  }
}
```

B.7 Data Logging

Logging the encoder values during initial testing was essential to the debugging of the system. We logged data by saving the values into an array while the system deployed, and then writing the values to the Serial monitor and/or a micro SD card after the system finished deploying. This minimized delays imposed on the system by logging data. The *logData* array and a few other variables were initialized as global variables before the **setup()** function:

```
int logPeriod = 5000; //micro seconds between each log entry (5ms)
String logHeaders[] = {"timeSinceDeploy","E1","E2","E3","E4","Shaft",
                       "DrillSwitch",
                       "S1Enable","S2Enable","S3Enable","S4Enable",
                       "S1Driving","S2Driving","S3Driving","S4Driving",
                       "IMU_LR","IMU_FB","IMU_LR_Accel","IMU_FB_Accel",
                       "IMU_Other_Accel"};
const int logDataRows = 401; //= MaxDeployTime / logPeriod  + 1
const byte logDataCols = 20; // Number of columns
float logData[logDataRows][logDataCols];
int logCounter = 0;          // Row number to save new data to
```

See the **Deploy** code above where the data is saved to the array during the system deploy. The **PrintLog()** function below saves the log to the micro SD card and printed it to the Serial monitor. Printing the data to the Serial monitor was convenient when initial tests were performed with the system tethered to the computer.

```
void PrintLog()
{
  Serial.println("Fall␣Direction:␣␣" + String(fallDirection));
  Serial.println("Data␣log␣of␣that␣run:");

  File dataFile = SD.open("walklog.csv", FILE_WRITE);

  for(byte i = 0; i < logDataCols-1; i++)
  {
    Serial.print(logHeaders[i]);
    Serial.print(",");

    dataFile.print(logHeaders[i]);
    dataFile.print(",");
  }
  Serial.println(logHeaders[logDataCols-1]);
  dataFile.println(logHeaders[logDataCols-1]);

  //Data
  for (int i = 0; i < logDataRows; i++) {
    for(byte j = 0; j< logDataCols - 1; j++)
      {
        if(j >= 14)
        {
          // Save the IMU columns with 5 decimal places
          Serial.print(logData[i][j], 5);
          dataFile.print(logData[i][j], 5);
        }
        else
        {
          Serial.print(logData[i][j]);
          dataFile.print(logData[i][j]);
        }

        Serial.print(",");
        dataFile.print(",");
      }
      // Save the IMU columns with more 5 places
      Serial.println(logData[i][logDataCols-1], 5);
      dataFile.println(logData[i][logDataCols-1], 5);

      delay(2);
  }
  dataFile.close();
  Serial.println("\nLog␣printing␣done!");
}
```

Bibliography

[1] World Health Organization, "Falls." https://www.who.int/news-room/fact-sheets/detail/falls, 2021. [Online; accessed 18-October-2021].

[2] C. for Disease Control and Prevention, "Keep on your feet: Preventing older adult falls." https://www.cdc.gov/injury/features/older-adult-falls/index.html, 2020. [Online; accessed 18-October-2021].

[3] B. Moreland, R. Kakara, and A. Henry, "Trends in Nonfatal Falls and Fall-Related Injuries Among Adults Aged $\geq$ 65 Years — United States, 2012–2018," July 2020.

[4] J.E. Walker and J. Howland, "Falls and Fear of Falling Among Elderly Persons Living in the Community: Occupational Therapy Interventions," 1991.

[5] C. Eustice, "Elderly Falls Tied to Canes and Walkers." https://www.verywellhealth.com/elderly-falls-tied-to-canes-and-walkers-2552063, April 2020. [Online; accessed 18-October-2021].

[6] E. Coyle, A. O'Dwyer, E. Young, K. Sullivan, and A. Toner, "Controlled breaking scheme for a wheeled walking aid," *IFAC Proceedings Volumes*, vol. 39, no. 21, pp. 21–25, 2006. IFAC Workshop on Programmable Devices and Embedded Systems PDeS 2006, Brno, Czech Republic, February 14-16, 2006.

[7] M. Azqueta-Gavaldon, I. Azqueta-Gavaldon, M. Woiczinski, K. Bötzel, and E. Kraft, "Automatic braking system and fall detection mechanism for rollators," in *Proceedings of the 6th International Conference on Bioinformatics and Biomedical Science*, ICBBS '17, (New York, NY, USA), p. 158–161, Association for Computing Machinery, 2017.

[8] Y. Hirata, S. Komatsuda, and K. Kosuge, "Fall prevention control of passive intelligent walker based on human model," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1222–1228, 2008.

[9] G. Lee, E.-J. Jung, T. Ohnuma, N. Chong, and B.-J. Yi, "Jaist robotic walker control based on a two-layered kalman filter," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3682 – 3687, 06 2011.

[10] Weiss, C.C., "Helite's airbag belt gives wearers a hip check." https://newatlas.com/helite-hipair-wearable-airbag/52867/, January 2018. [Online; accessed 11-October-2021].

[11] Tucker, Matthew, "Free fall tool harness and airbag system." https://www.behance.net/gallery/11232605/Free-Fall-Tool-Harness-and-Airbag-System, Oct 2013. [Online; accessed 11-October-2021].

[12] Weiss, C.C., "Helite readies wearable skiing airbag for 2014 winter olympics." https://newatlas.com/helite-airbag-skiing/26190/, February 2013. [Online; accessed 11-October-2021].

[13] Coxworth, Ben, "Airbag-equipped cycling vest instantly inflates when accidents happen." https://newatlas.com/helite-bsafe-airbag-cycling-vest/58050/, January 2021. [Online; accessed 11-October-2021].

[14] Hit-Air, Amazon, "Hit-air inflatable air vest sv2 model in black size s." https://www.amazon.com/Hit-Air-inflatable-vest-model-Black/dp/B00E40JJQ8/, 2021. [Online; accessed 11-October-2021].

[15] G. Shi, C. S. Chan, W. J. Li, K.-S. Leung, Y. Zou, and Y. Jin, "Mobile human airbag system for fall protection using mems sensors and embedded svm classifier," *IEEE Sensors Journal*, vol. 9, no. 5, pp. 495–503, 2009.

[16] J. Li, "Wearable and controllable protective system design for elderly falling," in *2020 6th International Conference on Mechanical Engineering and Automation Science (ICMEAS)*, pp. 187–194, 2020.

[17] G. Shi, C.-s. Chan, Y. Luo, G. Zhang, W. J. Li, P. H. W. Leong, and K.-s. Leung, "Development of a human airbag system for fall protection using mems motion sensing technology," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4405–4410, 2006.

[18] D. Trivedi, C. Rahn, W. Kier, and I. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, pp. 99–117, 10 2008.

[19] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.

[20] H. Lipson, "Challenges and opportunities for design, simulation, and fabrication of soft robots," *Soft Robotics*, vol. 1, no. 1, pp. 21–27, 2014.

[21] C. Majidi, "Soft robotics: A perspective—current trends and prospects for the future," *Soft Robotics*, vol. 1, no. 1, pp. 5–11, 2014.

[22] S. G. Nurzaman, F. Iida, L. Margheri, and C. Laschi, "Soft robotics on the move: Scientific networks, activities, and future challenges," *Soft Robotics*, vol. 1, no. 2, pp. 154–158, 2014.

[23] I. D. Walker, H. Choset, and G. S. Chirikjian, *Snake-Like and Continuum Robots*, pp. 481–498. Cham: Springer International Publishing, 2016.

[24] P. H. Nguyen, I. I. B. Mohd, C. Sparks, F. L. Arellano, W. Zhang, and P. Polygerinos, "Fabric soft poly-limbs for physical assistance of daily living tasks," 2019.

[25] A. D. Marchese, K. Komorowski, C. D. Onal, and D. Rus, "Design and control of a soft and continuously deformable 2d robotic manipulation system," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2189–2196, May 2014.

[26] G. Immega and K. Antonelli, "The ksi tentacle manipulator," in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 3, pp. 3149–3154 vol.3, 1995.

[27] W. McMahan, B. Jones, and I. Walker, "Design and implementation of a multi-section continuum robot: Air-octor," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2578–2585, 2005.

[28] E. W. Hawkes, L. H. Blumenschein, J. D. Greer, and A. M. Okamura, "A soft robot that navigates its environment through growth," *Science Robotics*, vol. 2, no. 8, p. eaan3028, 2017.

[29] Lowes, "Plastic stud support bracket." https://www.lowes.com/pd/Plastic-Stud-Support-Bracket/3574056, 2021. [Online; accessed 11-October-2021].

[30] POPLAY, Amazon, "Poplay thunder sticks, 100pcs inflatable sticks bam bam thunder sticks noise makers for sporting events." https://www.amazon.com/POPLAY-Thunder-Cheerleading-Inflatable-Noisemakers/dp/B01NA837I2, 2021. [Online; accessed 11-October-2021].

[31] TMORU, Amazon, "Tmoru 4 pcs pool inflatable sticks, 41in outdoor water games toy." https://www.amazon.com/TMORU-Inflatable-Sticks-Outdoor-Water/dp/B09CL12MVT, 2021. [Online; accessed 11-October-2021].

[32] Rockywoods SCA, "Heat sealable 200 denier oxford nylon - blue." https://www.rockywoods.com/Heat-Sealable-200-Denier-Oxford-Nylon-Blue, 2021. [Online; accessed 11-October-2021].

[33] HotJaw, Amazon, "Hotjaw kf-150cst 6in portable hand held heat sealer, 6 inch, black." https://www.amazon.com/Portable-Sealer-Model-KF-150CST-Storage/dp/B003HO30TE, 2021. [Online; accessed 11-October-2021].

[34] HATCHBOX, Amazon, "Hatchbox pla 3d printer filament, dimensional accuracy +/- 0.03 mm, 1 kg spool, 1.75 mm, black, pack of 1." https://www.amazon.com/HATCHBOX-3D-Filament-Dimensional-Accuracy/dp/B00J0ECR5I, 2021. [Online; accessed 11-October-2021].

[35] Ninjatek, "Ninjaflex 3d printer filament (85a)." https://ninjatek.com/shop/ninjaflex/, 2021. [Online; accessed 11-October-2021].

[36] SAINSMART, Amazon, "Sainsmart - tpu-blu-0.25kg1.75 sainsmart 1.75mm 250g flexible tpu 3d printing filament, dimensional accuracy +/- 0.05 mm (blue)." https://www.amazon.com/SAINSMART-Flexible-Printing-Filament-Dimensional/dp/B071SJ8SDQ/, 2021. [Online; accessed 11-October-2021].

[37] XHF, Amazon, "Xhf 4mm(wrapping range:1.5mm-10mm) spiral cable wrap spiral wire wrap cord for computer electrical wire organizer sleeve hose rohs black (dia 4mm length 20m)." https://www.amazon.com/dp/B07JNRL843, 2021. [Online; accessed 11-October-2021].

[38] Drive Medical, Amazon, "Drive medical 10210-1 deluxe 2-button folding walker with wheels." https://www.amazon.com/dp/B001HOM4U2/, 2021. [Online; accessed 11-October-2021].

[39] Teensy PJRC, "Teensy 4.1 development board." https://www.pjrc.com/store/teensy41.html, 2021. [Online; accessed 11-October-2021].

[40] Piedmont Plastics, "Aluminum composite material." https://www.piedmontplastics.com/products/acm, 2021. [Online; accessed 11-October-2021].

[41] Neiko, "Neiko 01148a hex allen power bit set, 11-piece metric sizes 1.5mm to 8mm — magnetic hex head bits — 3 quick release shanks — premium s2 steel — compatible with power drills and impact drivers." https://www.amazon.com/gp/product/B086RJW49Q/, 2021. [Online; accessed 28-October-2021].

[42] Super Lube, Amazon, "Super lube-21030 synthetic multi-purpose grease, 3 oz.." https://www.amazon.com/Super-Lube-21030-Synthetic-Grease/dp/B000XBH9HI, 2021. [Online; accessed 11-October-2021].

[43] RIDGID, Home Depot, "Ridgid 18-volt lithium-ion cordless brushless drill/driver and impact driver combo kit w/(2) 1.5 ah batteries, charger, and bag-r9603." https://www.homedepot.com/p/301853891, 2021. [Online; accessed 11-October-2021].

[44] Stepperonline OMC, "Dual shaft nema 17 bipolar 1.8deg 65ncm (92.3oz.in) 2.10a 3.36v 42x42x60mm 4 wires." https://www.omc-stepperonline.com/nema-17-stepper-motor/Dual-Shaft-Nema-17-Bipolar-18deg-65Ncm-923ozin-210A-336V-42x42x60mm-4-Wires.html, 2021. [Online; accessed 11-October-2021].

[45] CUI Devices, Digikey, "Amt102-v." https://www.digikey.com/en/products/detail/cui-devices/AMT102-V/827015, 2021. [Online; accessed 11-October-2021].

[46] Stepperonline OMC, "Dc electromagnetic brake 24v 0.25nm(35.4oz.in) for nema 17 stepper motor." https://www.omc-stepperonline.com/dc-electromagnetic-brake-24v-025nm354ozin-for-nema-17-stepper-motor-swb-01.html, 2021. [Online; accessed 11-October-2021].

[47] ANCIRS, Amazon, "4 pack 4mm flange coupling connector, rigid guide steel model coupler accessory, shaft axis fittings for diy rc model motors, high hardness coupling connector-silver." https://www.amazon.com/Coupling-Connector-Accessory-Fittings-Connector-Silver/dp/B07PDYV4P3, 2021. [Online; accessed 11-October-2021].

[48] Adafruit, "Adafruit aw9523 gpio expander and led driver breakout - stemma qt / qwiic." https://www.adafruit.com/product/4886, 2021. [Online; accessed 11-October-2021].

[49] Floureon, Amazon, "Floureon 2 packs 6s 22.2v 4500mah 45c lipo battery with xt60 plug." https://www.amazon.com/Floureon-4500mAh-Quadcopter-Airplane-Helicopter/dp/B01AW7CKLW, 2021. [Online; accessed 11-October-2021].

[50] HiLetgo, Amazon, "Hiletgo 2pcs digital voltmeter ammeter dc 100v 10a amp voltage current meter tester 0.56 inch 3 bits blue + red dual led display panel with connect wires." https://www.amazon.com/dp/B072BY4XZ7, 2021. [Online; accessed 11-October-2021].

[51] Pololu, "5v, 5.5a step-down voltage regulator d36v50f5." https://www.pololu.com/product/4091, 2021. [Online; accessed 11-October-2021].

[52] Pololu, "Pololu 12v, 15a step-down voltage regulator d24v150f12." https://www.pololu.com/product/2885, 2021. [Online; accessed 11-October-2021].

[53] HiLetgo, "Hiletgo 2pcs x9c104 digital potentiometer module 5v 40r-100k adjust bridge balance for arduino." https://www.amazon.com/gp/product/B01MTU9FSK, 2021. [Online; accessed 27-October-2021].

[54] SparkFun Electronics, "Sparkfun 9dof imu breakout - icm-20948 (qwiic)." https://www.sparkfun.com/products/15335, 2021. [Online; accessed 11-October-2021].

[55] HiLetgo, "Icstation txs0108e 8 channel logic level converter 3.3v 5v bi-directional high speed shifter for arduino raspberry pi iic i2c spi." https://www.amazon.com/gp/product/B06XWVZHZJ, 2021. [Online; accessed 27-October-2021].

[56] HiLetgo, "Grayhill 61c series optical encoder 61c22-01-04-02." https://www.digikey.com/en/products/detail/grayhill-inc/61C22-01-04-02/98658, 2021. [Online; accessed 27-October-2021].

[57] RENESAS, "Renesas x9c102, x9c103, x9c104 x9c503 digitally controlled potentiometer (xdcp) datasheet." https://www.renesas.com/us/en/document/dst/x9c102-x9c103-x9c104-x9c503-datasheet, 2021. [Online; accessed 18-October-2021].

[58] Mehran Maleki, "Interfacing x9c103s digital potentiometer module with arduino." https://electropeak.com/learn/interfacing-x9c103s-10k-digital-potentiometer-module-with-arduino/, 2021. [Online; accessed 24-October-2021].

[59] Adafruit, Github, "Adafruit_aw9523." https://github.com/adafruit/Adafruit_AW9523, 2021. [Online; accessed 24-October-2021].

[60] Last Minute Engineers, "Control stepper motor with drv8825 driver module & arduino." https://lastminuteengineers.com/drv8825-stepper-motor-driver-arduino-tutorial/, 2021. [Online; accessed 24-October-2021].

[61] Jacob Thompson, "Walker-continuum-robot-airbag." https://github.com/jakabo27/Walker-Continuum-Robot-Airbag, 2021.

[62] Vieyra Software, "Physics toolbox sensor suite." https://play.google.com/store/apps/details?id=com.chrystianvieyra.physicstoolboxsuite&hl=en_US&gl=US, 2021. [Online; accessed 15-October-2021].

[63] Wish, "K3 male straight hand straight foot body model mannequin skin color." https://www.wish.com/product/598c009b6b0da50c499b63e2?hide_login_modal=true, 2021. [Online; accessed 27-October-2021].

[64] Q. Zhang, H. Q. Li, Y. Ning, D. Liang, and G. R. Zhao, "Design and realization of a wearable hip-airbag system for fall protection," *Applied Mechanics and Materials*, vol. 461, pp. 667 – 674, 2013.

[65] R. Salomon, M. Lüder, and G. Bieber, "ifall - case studies in unexpected falls," in *2010 IEEE International Symposium on Industrial Electronics*, pp. 1645–1650, 2010.

[66] B.-J. Jo, Y. Lee, J. Kim, S. Y. Jung, D. Yang, J. Lee, and J. Hong, "Design of wearable airbag with injury reducing system," in *ICT4AgeingWell*, 2017.

[67] S. Sankaran, A. P. Thiyagarajan, A. D. Kannan, K. Karnan, and S. R. Krishnan, "Design and development of smart airbag suit for elderly with protection and notification system," in *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, pp. 1273–1278, 2021.

[68] L. Chen, D. Mao, and Q. Gao, *Design and Analysis of the Human Airbag*, pp. 133–138. Springer International Publishing, 01 2021.

[69] Z. Zhong, F. Chen, Q. Zhai, Z. Fu, J. P. Ferreira, Y. Liu, J. Yi, and T. Liu, "A real-time pre-impact fall detection and protection system," in *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1039–1044, 2018.

[70] J. Thompson and I.D. Walker, "Soft Continuum Robot Airbag Integrated with Passive Walker for Fall Mitigation." Submitted to IEEE RoboSoft Conference, 2022.

[71] Genuine Innovations, "Why do co2 cartridges get so cold?." https://www.genuineinnovations.com/blogs/learn-about-our-tech/why-do-co2-cartridges-get-so-cold, 2021. [Online; accessed 27-October-2021].

[72] Wikipedia contributors, "Carbon dioxide — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Carbon_dioxide&oldid=1048899486, 2021. [Online; accessed 11-October-2021].

[73] Discount Paintball, "Umarex 88 gram prefilled co2 cartridge high grade 2 pack." https://www.discountpaintball.com/umarex-88-gram-prefilled-co2-cartridge-2-pack.html, 2021. [Online; accessed 11-October-2021].

[74] JT Paintball, "Jt paintball 90gr pre-filled co2 air cartridges 2 pack." https://www.walmart.com/ip/JT-Paintball-90gr-Pre-Filled-CO2-Air-Cartridges-2-Pack/103804944, 2021. [Online; accessed 29-October-2021].

[75] Discount Paintball, "Tippmann 12 oz aluminum co2 tank for paintball." https://www.discountpaintball.com/tippmann-12oz-aluminum-co2-tank-for-paintball.html, 2021. [Online; accessed 11-October-2021].

[76] ANSgear, "Difference between co2 vs. compressed air for paintball?." https://www.ansgear.com/CO2_Vs_Compressed_Air_s/4835.htm, 2021. [Online; accessed 27-October-2021].

[77] Alex Smith, "What is the volume of an airbag when fully inflated?." https://rehabilitationrobotics.net/what-is-the-volume-of-an-airbag-when-fully-inflated/, 06 2021. [Online; accessed 18-October-2021].

[78] HELLA TECH WORLD - The Workshop's Friend, "Car airbag system." https://www.hella.com/techworld/us/Technical/Car-electronics-and-electrics/Car-airbag-system-3083/, 2021. [Online; accessed 18-October-2021].

[79] Tom Harbidd, "How much does airbag replacement cost." https://www.cashcarsbuyer.com/airbag-replacement-cost, 2020. [Online; accessed 18-October-2021].