Project Number: IQP MQF 2830

Ambulance Interior Storage Optimization

An Interactive Qualifying Project

submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements

for the degree of

Bachelor's of Sciences

By

William Jones, Josh Philippou, Nathaniel Sweet

Advised by Professor M. S. Fofana Date:

MIRAD Laboratory, Mechanical Engineering Department

Abstract

Inefficient storage in ambulances creates a dangerous environment for paramedics and patients. Currently items are held in bins located in overhead cabinets. When the cabinets are opened the items are susceptible to falling down as the ambulance is driven over bumps on the road or makes sharp turns. This can lead to the items becoming hazardous projectiles which can endanger both the paramedics and the patients. Furthermore, due to the limited space within an ambulance, paramedics often reach over patients or contort themselves in awkward positions in order to retrieve items. This leads to a host of injuries mostly relating to the back and shoulders which can leave paramedics unable to work or can cause long lasting damage. In the U.S., from 1999-2008 work related injuries of prehospital care, 255 paramedics accounted for 8,674 days spent away from work. This number does not take into account the work that is done by paramedics who are partially injured or are not completely healthy. This project attempts to address these problems by designing a vending machine to dispense consumables and medications. Pharmaceuticals and consumables currently held in cabinets are stored inside the vending machine in corkscrew holders. These items will be dispensed on request by rotating the holders and thereby dropping the items onto a conveyor belt leading to a single retrieval site close to the captain's chair in the patient compartment. Paramedics will request items by using a touch screen interface located within arm's reach of the captain's chair. The interface allows them to choose the specific item which can be easily located by first selecting what injury or condition they are trying to treat. The machine will displace which item is being requested through an interface showing the name of the item requested along with a description of the item. In doing so the machine ensures that patients receive the correct medication. This interface helps to identify and retrieve items from an original grouping of consumables and pharmaceuticals. The items may additionally be requested from a subgroup based on specific injuries or conditions. Extra items will no longer be opened up to the patient compartment, and the items are secured firmly in their individual holding mechanisms. There is no risk of additional items falling freely from the vending machine and creating the potential for injury. Furthermore, since all the items can be ordered and retrieved from a relatively unstrained position in the captain's chair using the touch screen, it will significantly reduce the stress being placed on paramedic's backs and shoulders.

Table of Content

List of Figures

List of Tables

Authorship

Chapter 1	Will Jones, Joshua Philippou, Nathaniel Sweet
Chapter 2 Introduction	Nathaniel Sweet
2.1	Will Jones
2.2	Joshua Philippou
2.3	Nathaniel Sweet
2.4	Nathaniel Sweet
Chapter 3 Introduction	Will Jones, Nathaniel Sweet
3.1	Joshua Philippou, Nathaniel Sweet
3.2	Will Jones
3.3	Nathaniel Sweet
Chapter 4	Nathaniel Sweet
SolidWorks Modeling	Joshua Philippou
Programming	Will Jones

Acknowledgements

We would like to thank Dr. Mustapha S. Fofana for his constant guidance throughout the project. Our team would like to thank UMass Emergency Services for allowing us to tour their facilities and for teaching us about the operations of the ambulance. Lastly we would like to extend our thanks to Neil Blackington of Boston EMS for his feedback and advice on our design.

CHAPTER 1: EMS and Life Saving Practices

Ambulances are necessary for the transportation of injured people to a place where they can receive proper medical attention. Ambulances are also important in the administration of preliminary medical care by emergency medical technicians (EMTs). The main composition of EMTs includes, basic life support, intermediate life and advanced life support or paramedics. EMTs are trained to perform a variety of medical treatments that save thousands of lives every year. While we depend on EMTs to provide us medical services, the ambulances they operate in often expose them to risk of injuries. In the current layout of most ambulances, medications and other supplies are stored in cabinets which are difficult to access. The difficulty stems from their placement in the ambulance. Additionally, while the location of the patient in the center of the vehicle is optimal for treatment, the patient may also act as an obstacle. This forces EMTs to either circumnavigate or reach over the patient in order to gain access to the equipment. EMTs overextend themselves, which leads to unnecessary strain on their body. Over time this strain can lead to injuries that may interfere with the EMT's ability to provide pre-hospital care. Furthermore, pharmaceuticals and consumables within the cabinets can move even at low vehicle speeds. Improper storage can lead to items falling out and become hazardous projectiles when cabinets are opened.

Inefficient storage in ambulances creates a dangerous environment for EMTs and patients. To rectify this problem the current storage in the ambulance is replaced with a vending machine. The vending machine must be able to fulfill several objectives. Namely, the design must securely store items so that they will not become dislodged. A customized dispenser is created to store and distribute pharmaceuticals and consumables. The vending machine should be reliable so that when an item is requested the EMT will receive that item in a timely manner. The dispenser is designed in such a way that it limits the chance of dropping unwanted items per request. This dispenser may be easily removed, and thereby allows quick restocking of the ambulance. A user interface is developed for the vending machine to respond fast and simple to navigate. Placement in the ambulance is very important for reducing

unnecessary movement. The items are dispensed to a location near the captain's chair for easy access. The vending machine contains an access panel to the motors that facilitates repairs. Limiting factors for the design include available workspace, power source and weight restrictions. The amount of storage space available in the ambulance is insufficient for most vending machine designs. To deal with this, a conveyor belt is employed. A conveyor belt allows for items to be delivered to a single location from anywhere in the machine. The power supply within the ambulance is already over consumed. To incorporate a machine that requires power, a separate source of power is needed. The vending machine has enough free space to accommodate an independent battery. The total weight of the vending machine is restricted such that the ambulance does not become overburdened or heavy. Lightweight materials used for the design of the vending machine.

The project examines the causes of injury to EMTs and patients in the ambulance rear compartment. The storage area of existing ground ambulances is modified to improve safety and efficiency. Chapter 2 presents the proposed improvements in ambulance design. The pharmaceuticals and supplies stored in the back of the ambulance are categorized. This chapter also includes an analysis of ambulance designs and the accessibility of supplies in the ambulance. The statistics of work related injuries to EMTs are presented and analyzed. An overview of the history, function and design elements of the vending machines is also discussed in Chapter 2. In Chapter 3, the proposed design and operation of the vending machine are presented. The software program, dimensions, location and functional elements of the vending machine are described. The software program allows the EMT's to select and retrieve pharmaceuticals from the machine. The software program and electrical system for the machine are discussed in this same chapter. Methods of evaluation consist of selected tests and a survey that verify the functionality of the vending machine. In Chapter 4, concluding remarks, an overview of the project is given with the software program and design elements reiterated. The impact of the design is also discussed in this chapter. Furthermore areas of improvement are identified. If this project is taken up by a future group, these areas should be viewed as important starting points in making the vending machine a viable option to be used in ambulances.

CHAPTER 2. EMS and Emergency Medical Care

The primary instrument EMTs use for providing emergency medical care is the ambulance. Ambulances have developed greatly over the last 150 years from simple horse drawn wagons to the technologically advanced vehicles you see in use today. Not only has the ambulance itself changed, but also its purpose in our society. Ambulances were originally used as means of transporting the injured or sick to clinic or hospital for further evaluation. With the growing desire for EMS personnel to administer prehospital care while transporting patients, it quickly became important for EMTs to receive more training. The requirement to carry pharmaceuticals and consumables onboard ambulances became important as well. The pharmaceuticals and consumables have different purposes and methods of administration. The choice of which drug or consumable to give to a patient requires careful consideration of the patient's condition. Within the ambulance, pharmaceuticals and consumables are currently stored in cabinets.

Despite all the advancements in ambulance technology and the ability to provide higher levels of emergency pre-hospital treatment than ever before, EMTs are required to put themselves at risk when working in the ambulance. EMTs sustain more injuries and are put at a higher risk of fatality than any other type of emergency care provider. Most of these injuries are incurred in the backs of paramedics and are a result of a poorly laid out storage system in the ambulance. The storage locations of pharmaceuticals force EMTs to move around the patient room and assume stressful positions in order to retrieve the necessary items to provide treatment. A storage system for pharmaceuticals and consumables modeled after vending machines would greatly reduce the risk of injury to EMTs and paramedics. Vending machines

incorporate advanced technology in their construction and their electrical system in order to provide requested items in the fastest and most reliable manner.

2.1 Ambulance Design and Improvements throughout History

The original and primary purpose of ambulances is to transport injured people to a place where they can receive further medical care: usually a clinic or hospital. Ambulances started their existence as simple horse drawn wagons, and over the years they have advanced to become the ambulances we see today across the world. With advances in the medical field, ambulances have taken on a new responsibility. They are now also tasked with providing preliminary emergency medical care to patients on the way to the hospital. The advent of the car as the primary means of transportation ushered in the modern era of ambulances. Ambulances use chassis of different models of cars as their base. Therefore the size, functionality, layout and design of an ambulance are largely based around which manufacturer and model the chassis comes from. While there has clearly been much improvement from the early wagon-drawn ambulances there is still plenty of room for development and innovation in the field.

2.1.1 Ambulances throughout History

During the American Civil War, William Hammond designed the first ever purpose-built ambulance. William Hammond, the surgeon general for the North, is considered to be the father of modern ambulance services[1]. The ambulance he designed was a simple horse drawn wagon and its only function was to transport wounded soldiers from the battle field. While its purpose was no different from the wagon, the motorized ambulance emerged during World War I as a faster means of transport and used a Ford Model T chassis which can be seen in Figure 1.

Figure 1: Ford Model T Ambulance

The Model T ambulance transported patients inside a wooden box attached to the back of the vehicle. The back room could transport three men laying down on cots or six people seated on them[2]. The 20 horsepower engine had a hand crank and was capable of achieving speeds over 45 miles per hour[3]. A Model T converted to an army ambulance had an average weight of between 1200 and 1500 pounds. Its lightweight made it possible for soldiers to lift the ambulance if it became stuck[4]. While being adequate for its time, this ambulance had several drawbacks to it that needed to be improved. The biggest problem was that the ambulance vibrated. The wooden bicycle tires had no suspension and were unable to suppress the vibrations of the often uneven roads. Another problematic feature was that the ambulance was constructed from wood and light metal which made the ambulance highly susceptible to weather, road conditions and degradation.

At the start of World War II the United States Army chose to use a Dodge chassis as the base of its new ambulances. There were many advantages to using the Dodge chassis versus the earlier Ford Model T ones. Dodge ambulances used steel plates in their frames which gave the ambulance increased protection from stray bullets. To overcome the issue of vibrations disturbing the ambulance, the Dodge ambulance had sixteen inch steel wheels with low pressure, all-weather pneumatic tires and hydraulic brakes. Additionally the Dodge ambulance

had a greater carrying capacity than the Ford Model T due to its modified box compartment. The compartment was capable of carrying four litters of wounded, seven people sitting or two litters of wounded and four people sitting. The purpose of ambulances underwent significant change during this time. Unlike the Ford ambulance, the Dodge ambulance also had a designated orderly for loading and unloading patients, preparing the ambulance, and administering first-aid as necessary. The underlying progress could be considered the precursor to the modern system of paramedics and EMTs we have today.

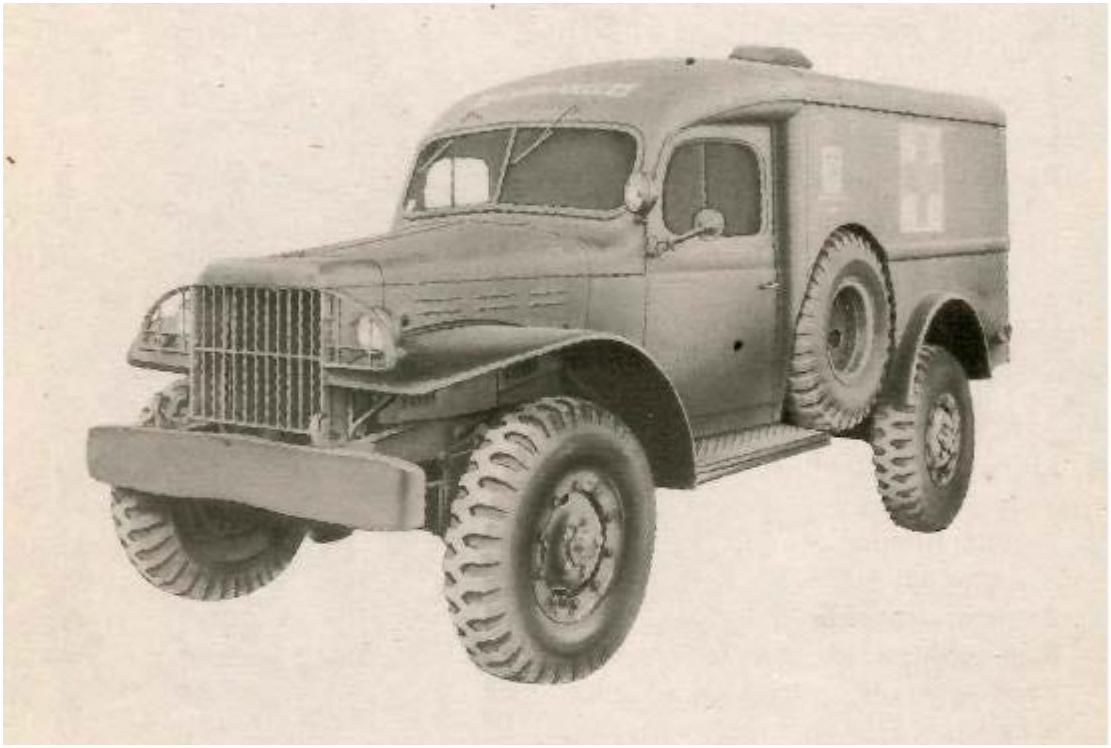

Figure 2: Dodge WC-4 3/4 on Ambulance

The Dodge 1942 WC-54 ¾ ton 4x4, shown in Figure 2, had further improvements and a wider frame that was lower to the ground than the earlier models. The ambulance's tires were an inch and a half wider than those of previous models. It also had a leaf-spring suspension system which was much better at combating rough terrain. A major design improvement in this model was double rear doors as opposed to an open back like earlier ambulances. Although this model was capable of travelling at speeds up to 54 miles per hour, it had extremely inefficient

fuel consumption and was only achieved 8 miles per gallon. Table 1 shows the general specifications of the ambulance.

Table 1: Dodge WC-54 3/4 ton General Specifications

General Data			
Crew		2	
Weight	Net	5,920 lbs	
	Payload	1,800 lbs	
	Gross	7,720 lbs	
Shipping Dimensions		790 cu ft.	105 sq ft.
Tires	Ply	8	
	Size	9.00 x 16	
Tread, center to center		64.75 in.	
Ground clearance		10.625 in.	
Electrical System		6 volts	
Capacities	Fuel (72 octane)	30 gal	
	Cooling system	8 qt	
	Crankcase (refill)	5 qt	
Brakes		Hydraulic	

This table contains some of the general data about the ambulance such as the breakdown of the weight. The net and the payload weights yield the gross weight. The table includes information on the tires used and the ground clearance that the ambulance has. The volume of fluids that the ambulance contains can be seen in the table and these include: the fuel capacity, the volume of fluid in the cooling system and the volume of fluid needed to refill the crankcase.

2.1.2 Conversion Era

A major period between World Wars I and II which was characterized by the conversion of available vehicles into ambulances became known as the Conversion Era. During this time most ambulances were made using hearses. Hearses already had a practical shape for ambulances. These conversions were conducted by hearse manufacturers as well as third party re-fitters. Some of the prominent companies that provided ambulances were the Automotive Conversion Corporation (ACC), Holcker Manufacturing Company, National Hearse and Ambulance Company, Owen Brothers, Pinner Coach Company, Rock Falls Manufacturing Company, Acme Coach Company, Henney Motor Company, Memphis Coach Company, Trinity Coach Company, A.J. Miller Company, and Weller Brothers Incorporated. The ACC Rescue-All, shown in Figure 3, was an ambulance model from the Conversion Era which made use of a refitted hearse.

Figure 3: ACC Rescue-All

The ACC Rescue resembled the ambulances seen today and showed several of the advances in ambulance technology that emerged during the Conversion Era. Most notable was the use of lights and sirens to make the ambulance's presence known. Furthermore the ambulance was

equipped with medical cabinetry, an attendant's seat, and a full sized stretcher. This allowed paramedics or EMTs to provide emergency medical care to patients while they are riding in the vehicle. Another change to ambulances that took place during the Conversion Era was an increased in the size of the passenger compartment. The Pontiac Model, manufactured by Memphis Coach and shown in Figure 4, has a raised roof to allow for more room in the patient compartment.

Figure 4: 1960's Memphis Coach Company Pontiac Model

The Pontiac ambulance was a 4-door sedan built for the US Air Force that was typically extended 24" but could be extended 36" as well. The unusually high roof allowed medics to treat patients with much greater comfort than other ambulances at the time. The larger size of the patient compartment also led to increased mobility for paramedics.

The last significant improvement of the Conversion Era was the introduction of temperature control in the passenger compartment of ambulances. Before this time ambulances were unable to regulate the temperature inside the compartment efficiently. Therefore the temperature within the ambulance compartment was very dependent on the outdoor temperature which could put patients in potentially harmful conditions. Internal heating of ambulances first appeared in the ambulances produced by Rock Falls Manufacturing. Figure 5 shows one of those heated ambulances.

Figure 5: Rock Falls Manufacturing "City Ambulance"

While heating of ambulances came relatively early in the Conversion Era, it was not until the 1940's that air conditioning was implemented in ambulances. Air-Conditioning was fairly expensive and was not in demand until after World War II when it became cheaper. The A.J. Miller Company offered ambulances with rooftop lights as well as heating and air-conditioning. Figure 6 shows one of the ambulances produced by A. J. Miller Company.

Figure 6: 1934 A.J. Miller Company Art Model

The EMS Systems Act of 1973 brought an end to the Conversion Era by implementing a strict set of guidelines for ambulances and medical care. These guidelines limited the freedom of design that existed in the early 1900's and made ambulances less unique and more standardized. This led to the Star-of-Life ambulance specifications that govern nearly every aspect of an ambulance design.

2.1.3 Modern Era

The EMS Systems Act of 1973 marks the beginning of the modern era. The era is defined by ambulance models that are classified as either Type I, II, or III according to the Star-of-Life Specifications[5]. The manufactures of modern ambulances include AEV, Braun, Crestline Coast, Demers, Excellance Inc, Frazer Inc, Horton, Leader, Life Line Emergency Vehicles, Marque Inc, McCoy-Miller Inc, Medix, MedTec, Miller Coach Company Inc, Osage Ambulance, Pl Custom Emergency Vehicles, Road Rescue, Taylor Made Ambulances, and Wheeled Coach[4]. The three types of ambulances each have different specifications that they have to satisfy. Type I ambulances have a cab chassis and a modular ambulance body. Type II ambulances are long wheelbase vans with integral Cab-Body. Type III ambulances are similar to Type I's but are a Cutaway van with integrated modular ambulance body. Types I and III have a subcategory AD (Additional Duty) for ambulances of a higher gross vehicle weight rating (GVWR). Type I ambulances are ambulances that have a cab chassis and a modular ambulance body like the Braun Super Chief seen in Figure 7. One of the chassis that Braun uses for the Super Chief is the Ford F-650. The ambulance has an overall length of 297.5 in, an overall width of 98 in, an overall height of 112 in, a module length of 170 in, and a module head room of 73 in[6].

Figure 7: Braun Super Chief

The Ambulance has a captain's chair, CPR seat, squad bench, Heat/AC system, air tank, suction pump, electrical outlets, inside/outside storage, and a slide out tray for two batteries. A constraint of Type I ambulances is that the GVWR has to be between 10,001 and 14,000 pounds. If the GVWR goes beyond 14,000 the ambulance is labeled as a Type I – AD.

Type II ambulances are long wheelbase vans with integral Cab-Body. The GVWR of Type II ambulances have to be between 9,201 and 10,000 pounds. A good example of a Type II ambulance is the AEV Sprinter 2500 Van TraumaHawk[7]. The ambulance chassis is a Sprinter M2CA144 2500 with a 3.0 V6 Mercedes Turbo Diesel engine. The ambulance contains a captain's chair, squad bench Heat/AC system, suction pump, electrical outlets, and interior storage. This ambulance is much simpler in design than the Braun Super Chief due to the limits of the chassis. Figure 8 shows diagrams of the ambulance interior from the left side, the right side, and the back. A Type II ambulance is a smaller ambulance and does not have the capacity of a Type I or III.

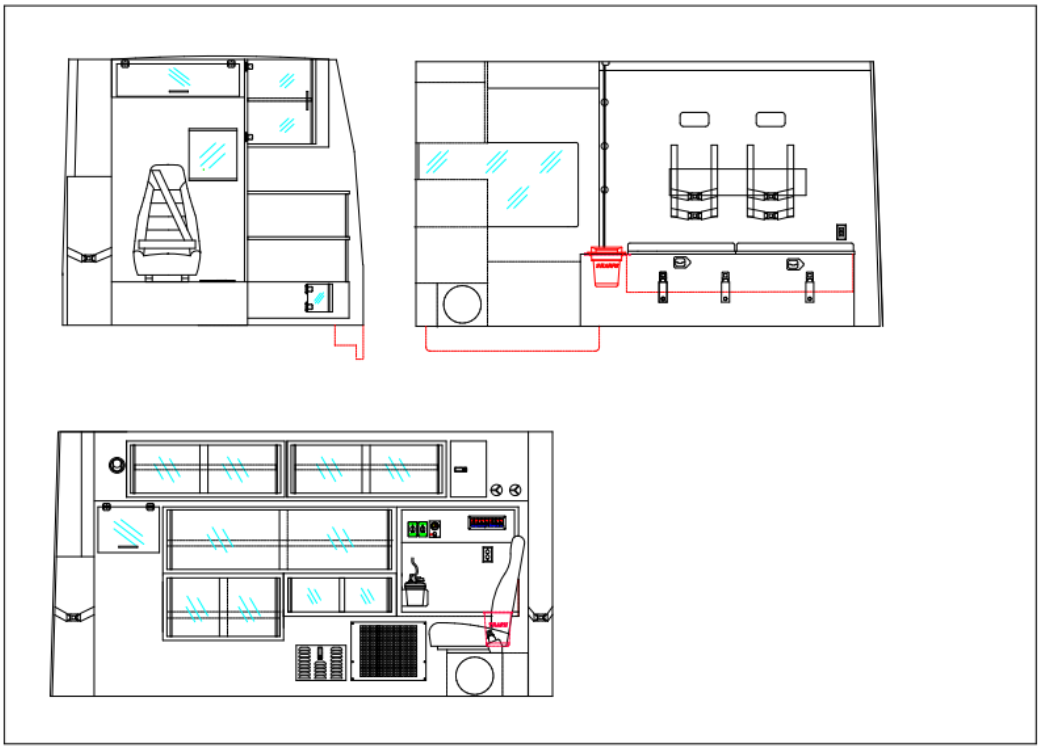

Figure 8: AEV Sprinter 2500 Van TraumaHawk.

The last type of ambulance, the Type III, is similar to Type I ambulances but is a cutaway van with integrated modular ambulance body. A Type III ambulance can have a GVWR of between 10,001 and 14,000 pounds. An example of a Type III ambulance is the Braun Express, of which one of the available chassis is the Ford F-350. Figure 9 shows an external view of the ambulance while Figures 10 and 11 show the diagrams of the ambulance.

Figure 9: Braun Express

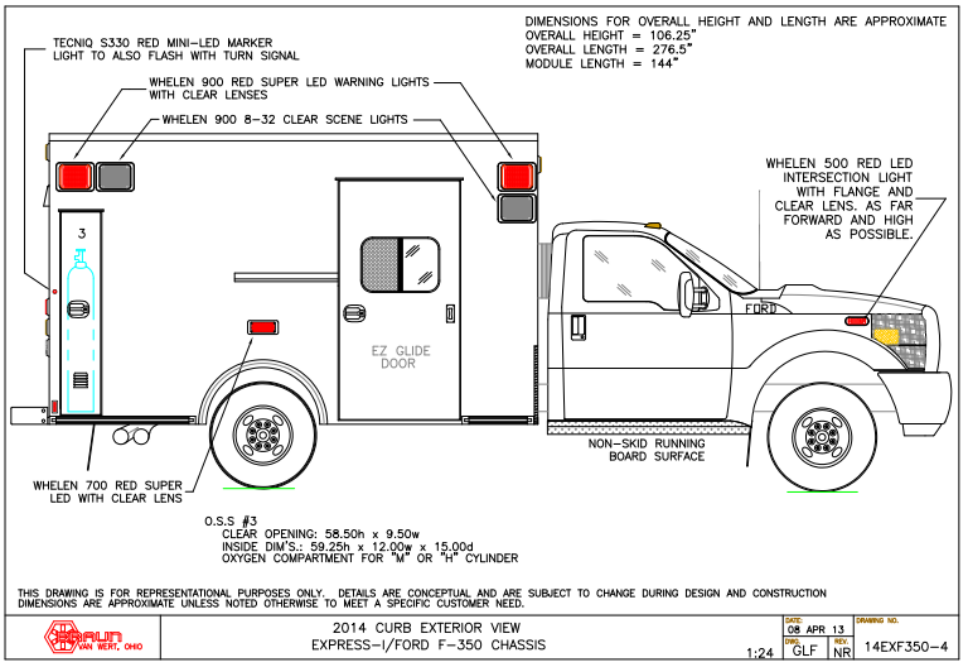

Figure 10: Braun Express Side View

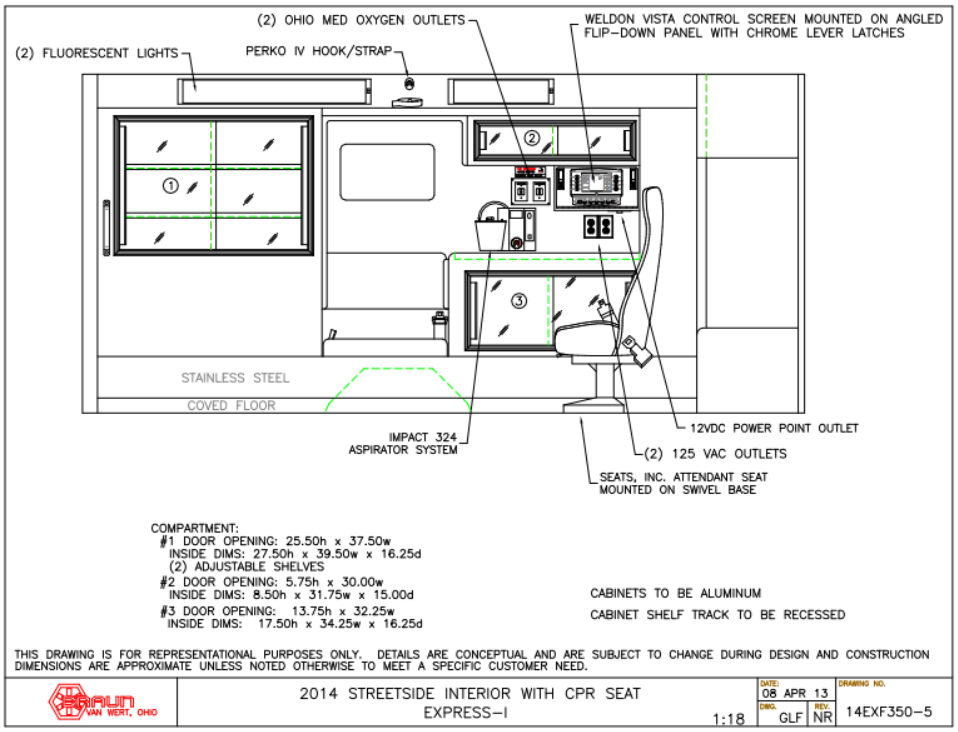

Figure 11: Braun Express Left Interior View

The Braun Express has an overall length of 276.5 in, an overall width of 95 in, an overall height of 104.5 in, a module length of 144 in, and a module head room of 68 in[6]. The ambulance has a

captain's chair, CPR seat, squad bench, Heat/AC system, air tank, suction pump, electrical outlets, and inside storage. A constraint of Type III ambulances is that the GVWR has to be between 10,001 and 14,000 pounds. If the GVWR goes beyond 14,000 pounds the ambulance is labeled as a Type III – AD.

2.2 Studies of EMS and Ambulances

Emergency medical services (EMS) have become an important part of the United States healthcare system. It is imperative that the emergency medical technicians (EMTs) running and operating ambulances be protected from any unnecessary risks and injuries. Data from the United States Bureau of Labor Statistics shows that this is unfortunately not the case. EMTs are at a higher risk of severe injury than nurses and other medical care providers. These EMTs are not only getting injured but they are missing work as a result of injuries. During ambulance crashes they are more at risk of injury and death than other emergency responders. Previous studies have shown the importance of ergonomic design in ambulances. Ergonomic analysis of the interior rear compartment of ambulances helps identify the key areas that need attention. Primarily it is difficult for EMTs to reach to equipment, pharmaceuticals and consumables, and the patient while buckled in a proper sitting position.

2.2.1 Statistics on Ambulance Related Injuries

Work performed by EMTs is often strenuous and can result to injuries and missed work. As seen in Tables 2 and 3 there were reported 5170 nonfatal work related injuries to EMTs in the United States in 2004. These were injuries that were severe enough to require time spent away from work to recover. Those 5170 injuries were further separated by the body part affected by the injury. The body parts are divided by the columns of the table. There are five main categories: head, neck, trunk, upper extremities, and lower extremities. These categories are further split into specific body parts. Each row represents the work related injuries of a different occupation. The number in each cell corresponds to the number of workers that sustained that type of injury while working and required days off. Of those injuries, 2950 (57%) were to the backs and shoulders of EMT's, 190 were neck and head injuries, 720 to the upper extremities, and 600 to the lower extremities. Body systems denoted injuries to internal organs and accounts for 170 of the 5170 injuries. For occupations like firefighters that do not have a lot of injuries resulting in days from work much of the table is left blank. Data is only presented if the number of injured exceeds ten. All numbers are rounded down to a multiple of ten.

Table 2: Nonfatal Occupational Injuries by Selected Parts of Body Affected, 2004[8]

Occupation	Number in Private Industry	Part of Body Affected by Injury or Illness					
		Head		Neck	Trunk		
		Total	Eyes		Total	Back	Shoulder
EMTs and Paramedics	5170	60	-	130	3140	2410	540
Firefighters	40	-	-	-	-	-	-
Police	110	-	-	-	-	-	-
Nurses	20500	880	240	680	9760	7260	1400
Physician and Surgeon	250	-	-	-	80	70	-
Construction Workers	37930	3490	2130	540	12190	8160	1780

Table 3: Nonfatal Occupational Injuries by Selected Parts of Body Affected, 2004, continued[8]

Occupation	Number in Private Industry	Part of Body Affected by Injury or Illness								
		Upper Extremities				Lower Extremities			Body Systems	Multiple Body parts
		Total	Finger	Hand	Wrist	Total	Knee	Foot		
EMTs and Paramedics	5170	720	140	130	280	600	280	60	170	350
Firefighters	40	-	-	-	-	-	-	-	-	-
Police	110	-	-	-	-	50	-	-	-	-
Nurses	20500	2470	640	310	770	3290	1380	570	350	2800
Physician and Surgeon	250	100	-	-	40	-	-	-	-	-
Construction Workers	37930	9000	4060	1940	1100	10050	3040	2260	350	2110

The number of injured nurses that required days away from work in 2004 was higher than the number of EMTs and paramedics, but this does not mean that nurses are at higher risk of injury. There were roughly 2.7 million nurses working in 2012 while there were only 239,100 EMTs and paramedics working in 2012[9,10]. For every one EMT or paramedic there are at least ten nurses, so even though the number of nurses injured is higher, the percentage of nurses injured is lower. Both of these jobs require the worker to perform medical treatment, but the EMTs are getting injured more frequently. Physicians and surgeons are also medical professionals but they are not injured nearly as much as EMTs or nurses. Since nurses and EMTs have to perform a similar wide variety of tasks in their jobs the difference in injury rates is probably linked to their work environments. Although firefighters and police officers are both part of emergency services they are severely injured less often than EMTs and paramedics. Possible reasons for this are the kind of work they do and the environment that they must do it in.

In 2011, the number of work related injuries in most of the listed occupations decreased. American EMTs only sustained 4900 work related injuries as opposed to the 5170 in 2004.The specific number can be seen in Tables 4 and 5. The tables are set up where the columns denote location of the injury and rows indicate the occupation. The second column from the left is the total number of injured within the occupation. Between 2004 and 2011, the total number of injured nurses that required day away from work increased by about 1500. The number of nurses is projected to grow 19% between 2012 and 2022, so this increase represents a slight decrease in the percent of injured nurses. The number of injured construction workers decreased by half in those seven years. Firefighters remained the same. Physicians and surgeons went down to only slightly above 200 injuries. Police officers had a slight increase, but like the nurses this slight increase is balanced out by the increase in officers.

Once again the back and shoulders were the most affected regions of the body, accounting for 54% of the injuries in EMTs. Back injuries were the most prevalent and shoulder injuries were the second. These are the main areas of concern as lifting pharmaceuticals and consumables from the storage places a lot of stress on these parts of the body. The stress is increased if the EMTs are bending over a patient to reach something. Having to lean over to treat a patient also put a strain on their lower and mid back muscles. All these stresses can lead to slight injuries that become increasingly severe over time. The injury can become severe enough to warrant time off from work.

Table 4: Nonfatal Occupation-al Injuries by Selected Parts of Body Affected, 2011[11]

Occupation	Number in Private Industry	Part of Body Affected by Injury or Illness									
		Head		Neck	Trunk		Upper extremities				
		Total	Eye		Total	Back	Total	Shoulder	Arm	Head	Wrist
EMTs and Paramedics	4900	230	110	40	2420	2200	870	460	140	120	90
Firefighters	40	-	-	-	-	-	-	-	-	-	-
Police	150	-	-	-	30	30	20	-	-	-	-
Nurses	22150	990	150	660	7630	6720	5320	1800	720	1590	810
Physicians and Surgeons	220	20	-	-	30	20	70	-	-	20	-
Construction Workers	18600	1080	590	160	4190	2890	6160	1040	930	3640	430

Table 5: Nonfatal Occupational Injuries by Selected Parts of Body Affected, 2011, continued[11]

Occupation	Number in Private Industry	Part of Body Affected by Injury or Illness					
		Lower Extremities				Body Systems	Multiple Body Parts
		Total	Knee	Ankle	Foot		
EMTs and Paramedics	4900	850	330	270	80	30	450
Firefighters	40	-	-	-	-	-	-
Police	150	60	40	-	-	-	20
Nurses	22150	3990	1750	920	510	330	3020
Physician and Surgeon	220	50	-	20	-	20	30
Construction Workers	18600	4820	1270	1270	1160	500	1580

Tables 6 and 7 encompass injuries that are resulted in time off by their source instead of injury location. The source of injury is listed in the columns while the occupation is listed in the rows. If the person was injured do to something that they did it falls under the category Person-Injured or Ill Worker, while injuries caused by another person are listed under Person-other than Injured Worker. The source of injury can vary from occupation to occupation. Parts and materials for instance could mean a pharmaceutical or consumable for EMTs or a piece of wood for a construction worker. Injuries caused by chemicals and hand tools were left out of the table as they only affected one or two of the occupations or were very occupation specific.

Most of EMTs injuries are caused by patients, but a lot are also injured because of the posture or motion required to do their job. Of the 4900 EMT injuries in 2011, roughly 40% were caused by the patients, while 10% were caused just by the EMTs' posture and motions. Almost 20% of the injuries came from unspecified sources, and the remaining 30% was mostly because of vehicles and floors. Nurses and EMTs have very similar division of injuries by source. The percentages matched up in most cases other than vehicle. The EMTs should have a higher percent caused by vehicle since their job requires being on the roads. About 10% of the nurses are also being injured as a result of their posture or motions. The similarities suggest that the work environment is the biggest factor of injury source. The percent of EMTs requiring days off because of a work related injury is higher than the percent of nurses, but they are being injured by the same things. The higher percentage of injured shows the increased risk caused by their work space, whiles the similar distribution of causes and affected body parts denotes similar working tasks.

It is difficult to compare injury source between EMTs and other occupations. Firefighters, police officers, and physicians and surgeons do not have enough injuries. The table is left mostly blank for these occupations because there needed to be at least 10 injuries a category. There was enough data on construction workers to be able to compare however. The difference in source distribution is clearly caused by the difference in work being done. Construction workers do not have patients, which account for 40% of EMTs injures, so their injuries are divided between the other categories. Similarly EMTs do not have to rely heavily of hand tools so the injuries those tool cause to construction workers cannot be compared. The injuries caused by patients to EMTs and the injuries caused by hand tools to construction workers do not match up making the distribution into all the other categories unequal as well.

Table 6: Nonfatal Occupational Injuries and Illnesses by Selected Source, 2011[12]

Occupation	Number in Private Industry	Source of Injury or Illness			
		Containers	Furniture and Fixtures	Parts and Materials	Floors
EMTs and Paramedics	4900	100	90	50	520
Firefighters	40	-	-	-	-
Police	150	-	-	-	30
Nurses	22150	420	1040	300	4440
Physician and Surgeon	220	-	-	-	90
Construction Workers	18600	730	160	4640	1580

Table 7: Nonfatal Occupational Injuries and Illnesses by Selected Source, 2011, continued[12].

Occupation	Source of Injury or Illness					
	Vehicles	Person-Injured or Ill Worker		Person-other than Injured Worker		All other sources
		Total	Worker Motion or Position	Total	Patient	
EMTs and Paramedics	520	470	450	2150	1950	960
Firefighters	-	-	-	-	-	-
Police	30	20	20	40	-	20
Nurses	970	2420	2370	9020	8750	2920
Physician and Surgeon	-	30	30	40	40	-
Construction Workers	1150	1500	1480	30	-	4280

2.2.2 Statistical Study of a Sample Area

Between 1999 and 2008 there were 255 lost time injuries that account for 8674 days spent away from work in Austin Travis County EMS. On average those 255 people lost 34 days of work because of a work induced injury. Figure 12 shows the percentage of EMT's in the Austin Travis County EMS injured during the nine year observation period. The y-axis is the percent of EMTs that were injured during a given year, and the x-axis indicates the year that the data is for. There are four different categories that injuries are broken up into. The red line indicate injuries that are from strains and sprains, the green line donates injuries to the EMTs back, neck and knees, the purple line for injuries only in the EMTs back and the light blue line is for injuries caused by lifting and lowering stretchers. The dark blue line shows the total percentage of EMTs that were injured in a given year[13].

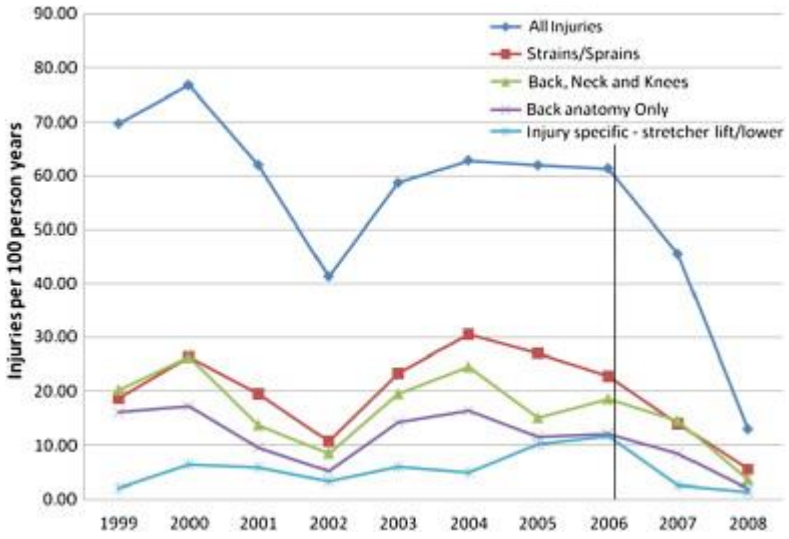

Figure 12: Injury Rate per 100 Fulltime Equivalents per Year[13]

The study in the Austin Travis County EMS also reported the percent of injured EMTs that required days away from work, Figure 13. The injuries are broken into the same categories as the above graph with the exception of lifting and lowering stretchers which was omitted. The y-axis was fitted to the data and is only out of 18 instead of the 90 in the above graph. Figure 12 shows that around 40 to 80% of EMTs sustained injures during the nine year study, while Figure 13 shows that between 7 and 15% of EMTs required days off from work as a result of these injuries. This shows that a large percentage of EMTs are still working while injured, which may exacerbate their injuries.

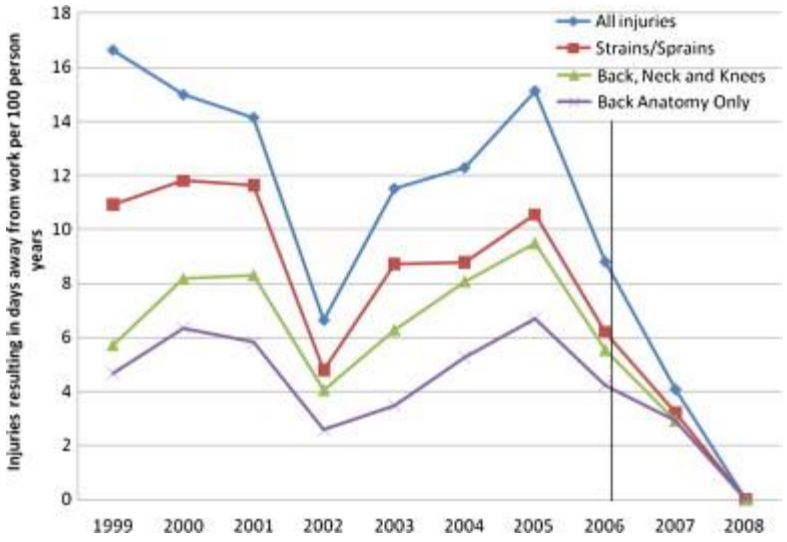

Figure 13: Rate of Lost Workday Injuries per 100 Full-Time Equivalents[13]

It is estimated that 55 percent of EMTs are injured every year within the Austin Travis County EMS. According to the study this number should be similar to EMS of other urban areas[13].

Occupants of ambulance crashes are more often injured or killed than those within other emergency response vehicles. These fatalities occur in 0.2% of ambulance crashes from 1988-1997 which is almost double that of police cars and fire trucks. 28% of ambulance occupants suffer an injury during a crash which is equal to police cars while more than double fire trucks rates. Table 8 shows the percentages of injuries during crashes based on data from the National Highway Traffic Safety Administration's Fatality Analysis Reporting System and the General Estimates System between 1988 and 1997. In crashes occurring during emergency calls, 13.4% of restrained back-seat occupants suffer non-incapacitating injuries, 0.1% suffer incapacitating injuries, and 0.17% are killed. When compared to unrestrained rear occupants, these numbers rise drastically. During the same period, 19.5% of unrestrained back-seat occupants sustained non incapacitating injuries, 11.7% were incapacitated, and 0.4% were killed during crashes. While restrained occupants suffered almost no incapacitating injuries during the crashes, almost 12% of unrestrained occupants were knocked unconscious. Furthermore, unrestrained rear occupants are 1.5 times as likely to be injured while still retaining consciousness, and twice as likely to be killed. During routine trips, only 2% of restrained rear occupants suffered an injury or fatality compared to an overwhelming 36% injury or fatality rate for unrestrained occupants in the case of a crash[15].

Table 8: The Injury Severity of Ambulance Occupants Involved in Crashes: 1988-1997[15]

		Injury severity (frequency)				Total occupants	Total vehicles involved in crashes
		No injury	Possible/non-incapacitating	Incapacitating	Fatal		
Emergency call							23,474
Front-seat	Restrained	27,873 (88.28)	3,305 (10.47)	390 (1.24)	4 (0.01)	31,572	
	Unrestrained	2,479 (79.92)	607 (19.57)	13 (0.42)	3 (0.10)	3,102	
Seated in the back	Restrained	3,071 (86.34)	475 (13.35)	5 (0.14)	6 (0.17)	3,557	
	Unrestrained	3,094 (68.38)	882 (19.49)	531 (11.73)	18 (0.40)	4,525	
Routine trip							12,492
Front-seat	Restrained	11,585 (86.03)	1,562 (11.60)	313 (2.32)	7 (0.05)	13,467	
	Unrestrained	829 (66.16)	198 (15.80)	220 (17.56)	6 (0.48)	1,253	
Seated in the back	Restrained	1,600 (97.92)	26 (1.59)	0 (0.00)	8 (0.49)	1,634	
	Unrestrained	1,717 (64.21)	741 (27.71)	197 (7.37)	19 (0.71)	2,674	
Total		52,248 (84.57)	7,796 (12.62)	1,669 (2.70)	71 (0.11)	61,784	35,966

Values shown in parentheses are percentages.

2.2.3 Reviewing Ambulance Design

One study set out to evaluate the efficiency of ambulances in the United Kingdoms. The ambulance being studied can be seen in Figures 14 and 15. The important issues that it considered were how the ambulance layout affected efficiency of the care given and the safety of the Emergency Medical Technicians (EMTs) and their patients. The metric for efficiency was how many items the EMTs could reach from the two different seats in the rear of an ambulance[16]. The study was prompted by high rates of musculoskeletal problems seen within the EMTs[17] [18] [19] [20]. Factors such as high vibration, sitting posture, and sporadic heavy lifting after long periods of limited motion were identified as the key reasons for musculoskeletal injuries. Musculoskeletal injuries if severe enough can require days or even months off to properly recover[16]. Ergonomics is the science of maximizing efficiency, so an ergonomic evaluation of ambulances' rear compartment should identify the key challenges of the current design. While there are previous studies on postures of the EMTs, the researchers determined them to be too limited. They were limited because they did not focus on which activities required the EMTs to assume the most harmful postures. A study done in Canada surveyed EMTs and found that the most physically demanding tasks were cardiopulmonary resuscitation, loading the stretcher, accessing the patient and equipment, and working from the seat in general[18].

Figure 14: United Kingdom Ambulance being studied[16]

Figure 15: Inside of Ambulance[16]

Doormaal et al. determined that about a quarter of EMTs postures need correction to lower unnecessary stress during non-emergency calls, and over half in emergency calls[20]. Evaluations of the assumed postures were done visually while various tasks were performed to determine the risk of injury. Combined with how frequently those tasks were being performed gives a basis for which areas in the ambulance need the most attention. Table 9 shows how frequently tasks were performed during the study, and which tasks require postural correction. AC stands for action category and is the part of the postural analysis system used during the study. An AC of 1 means no action is required and an AC of 4 means immediate correction is necessary[16]. Most of the tasks required no postural correction or a slight correction. Transferring the patient was the only task where over fifty percent of postures need immediate correction.

Table 9: Ovako Working Posture Analysis System action category for clinical tasks[16]

Task	*N*	AC=1 (%)	AC=2 (%)	AC=3 (%)	AC=4 (%)
Writing on clipboard	297	94	6	—	—
Interaction with patient	292	70	30	—	—
Idle	194	100	—	—	—
Interaction with carer	74	82	16	—	2
Accessing equipment[a]	56	41	54	3	2
Using cardiac monitor[a]	55	60	40	—	—
Using pulse-oximeter	51	67	27	4	2
Loading/unloading patient	45	62	27	7	4
Other[a]	38	47	50	3	—
Cannulation/drug administration[a]	31	32	68	—	—
Blood pressure check[a]	30	57	43	—	—
Blood glucose check[a]	26	58	42	—	—
Rubbish/sharps disposal[a]	23	9	91	—	—
Oxygen administration[a]	17	59	41	—	—
Non-specific motion	16	75	25	—	—
Talking to driver/ on phone[a]	12	25	75	—	—
First aid treatment[a]	11	57	43	—	—
Transferring patient[a]	9	11	22	56	11
Listening with stethoscope[a]	8	50	50	—	—

The efficiency of the layout was determined by a range analysis and a link analysis from the two main sitting positions. The range study creates a two dimensional representation of the EMTs range while sitting upright and being restrained in the chairs (Figure 16). The arch around the seats represents the range that an EMT of average height can reach from a proper sitting position. From seat B almost nothing is within reach of the EMTs. The EMTs in this seat should

not be able to reach the patient to administer treatment. This means that the EMT has to sit forward and most likely remove part or all of their seat belt to increase their range. Seat A can more easily reach the equipment but cannot reach the patient's torso. This equipment includes the pulse oximeter, defibrillator, portable oxygen, and the portable Entonox. The suction, rubbish, sharps and blankets are not within reach of either of the EMTs. Disposing of used pharmaceuticals and consumables is difficult as the EMTs must reach over the patient. It is very likely that the EMTs must stand or completely remove their seat belts to reach either to rubbish or sharps bins. The link analysis, seen in Figure 17, shows what equipment was accessed from which seat and how frequently the items were used. Items that were frequently used by the EMT in seat A will have more lines connecting the item and seat A. When an item is more frequently used on a patient there are more lines connecting the stretcher with the item.

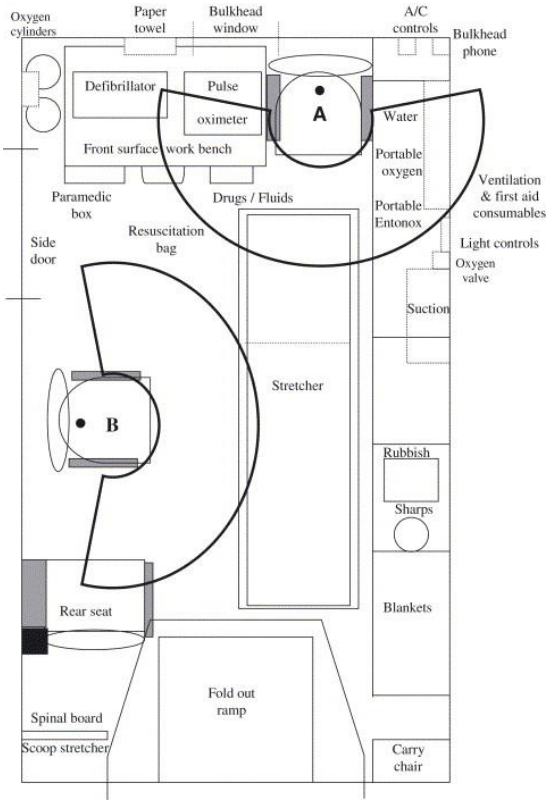

Figure 16: Range Diagram of Average size EMT Sitting upright with a Seatbelt[16]

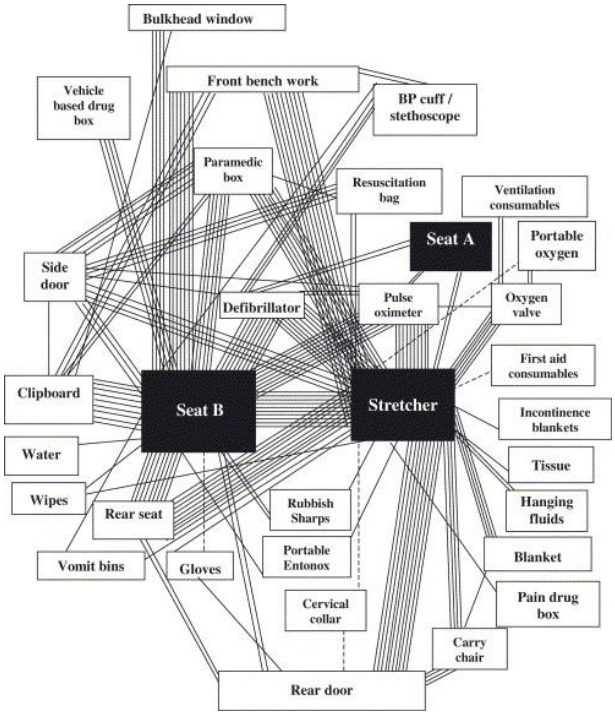

Figure 17: Schematic Link Analysis[16]

The study found that the EMTs could not reach important areas, like the disposal, or perform treatment while buckled with a correct sitting posture. This led to EMTs having to at least lean forward if they wanted to administer any form of treatment and in many cases they had to remove their seat belts to do so. The lack of restraints to keep the EMTs firmly in their seats during sudden shifts in the ambulance is very dangerous, and puts both the EMT and the patient at risk of injury. The study recommended a different layout of the ambulance that facilitated the retrieval of equipment easier and safer for the EMTs.

There were several limitations on this study that were important and should have been taken into consideration. First the level of severity of the calls that they witnessed was heavily weighted towards non-life-threatening calls. This explains why the most used equipment and treatments pertained to checking patient's heart rates, blood pressure, and blood oxygen saturation. The full scope of what EMTs have to do during a life-threatening call is still largely undocumented and analyzed. The second limitation is how the posture analysis was done. The researchers had to rely on observational analysis because any equipment that would read a more accurate posture might interfere with the EMT's work. Because of these limitations the researchers did not think that they had enough data to effectively redesign the ambulance and therefore gave no helpful conclusion[16].

The range analysis is a great way to better show where EMTs can easily reach from different positions. Conducting a range analysis from sitting and standing positions along with allowing one or two steps would more accurately show which areas of storage are difficult or dangerous to reach. Those areas would then become the main focus of the redesign. If some areas are much easier to reach than others, improvements for them might not be relevant or desired. Performing a link analysis might help visualize a small set of items, but is not practical when looking at storage. There are a lot of pharmaceuticals and consumables being stored and the diagram created from the link analysis would be too cluttered and difficult to read.

2.3 Emergency Medical Care

In 2014 Massachusetts was ranked second in the nation by the American College of Emergency Physicians (ACEP) in terms of its emergency medical care system, second only to the District of Columbia[21]. Even though it is ranked highly compared to the rest of the Nation, it received a grade score of B^-. The United States emergency care system as a whole is ranked a distressing D^+ so there is much room for improvement not only in Massachusetts but in the entire country (see Table 10). ACEP has chosen the five categories of access to emergency care, quality and patient safety environment, medical liability environment, public health and injury prevention and disaster preparedness in order to better evaluate each state's emergency medical care system. Access to emergency care is judged based on the population's access to providers, access to treatment centers, financial barriers to do so and hospital capacity. Quality and patient safety environment is split up into state run systems and institutional systems. Each state's medical liability environment grade is made up of the legal atmosphere, the availability of insurance and by tort reform. The public health and injury prevention category is graded based on cases of traffic safety and drunk driving, immunization, fatal injuries, health and injury prevention efforts and health risk factors. And the final category is disaster preparedness. This category is composed of financial resources, state coordination in the case of an emergency, hospital capacity and personnel to deal with a disaster.

Table 10: Emergency Care Report Card for Massachusetts and the United States

	2009	Massachusetts		United States	2014	Massachusetts		United States
		Rank	Grade	Grade		Rank	Grade	Grade
Access to Emergency Care		3	B	D-		4	B	D-
Quality & Patient Safety Environment		6	A	C+		5	B+	C
Medical Liability Environment		33	D	C-		40	D-	C-
Public Health & Injury Prevention		1	A	C		1	A	C
Disaster Preparedness		19	B	C+		20	C	C-
Overall		1	B	C-		2	B-	D+

In the table, the ranks and grades for Massachusetts are given as well as the grades for the United States in both 2009 and 2014. As the table shows, Massachusetts excels in public health injury prevention, quality and patient safety environment. Access to emergency care in Massachusetts is good but lacks greatly in disaster preparedness and medical liability environment. The emergency medical systems for Massachusetts and for the country as a whole have gotten lower grades in 2014 than in 2009 in several of the categories. Each red box in the table indicates that the either the grade or the rank has decreased from 2009 to 2014. The complete ACEP grading criteria for emergency medical care is broken down into five categories (see Appendix A along with the scores for the United States). Appendix B contains the scores given to Massachusetts for its emergency program.

Figure 18 shows which states received what grades for the category of quality and patient safety environment. While many states provide a decent level of emergency care, far too many others, particularly in the central United States, do not do enough for their citizens. While Maryland, Utah, Pennsylvania and North Carolina fared the best and they all received A

grades in this category, ten different states, an increase from just three in 2009 all received F's for their abysmal quality and patient safety environment programs.

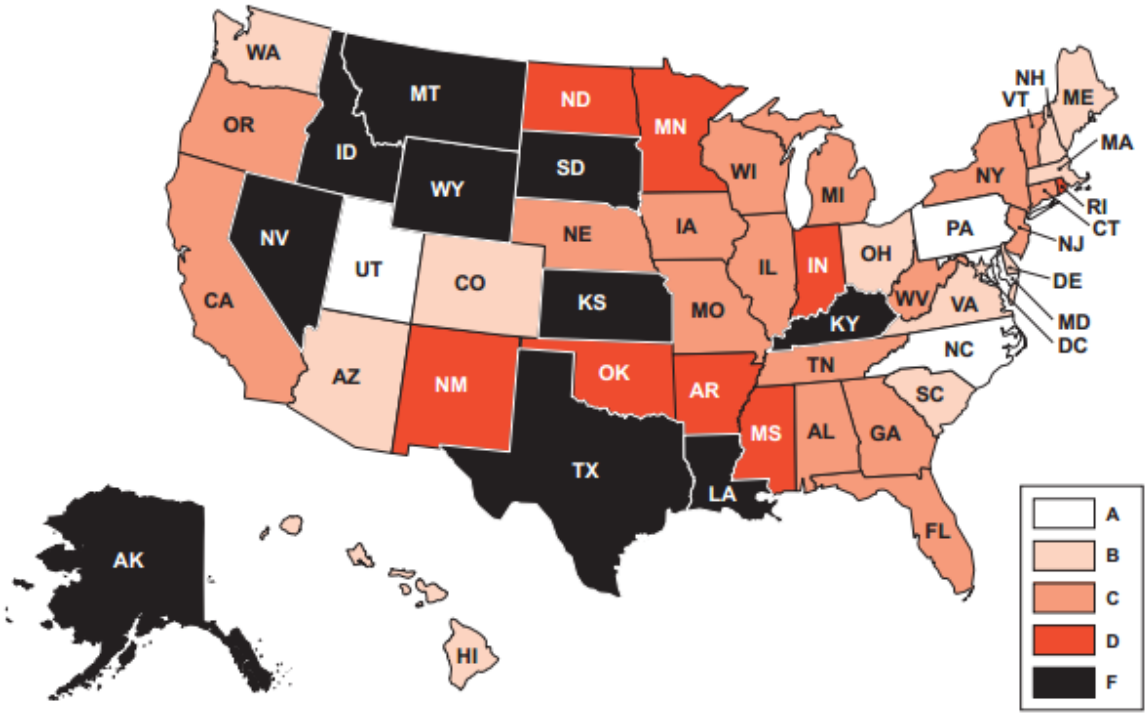

Figure 18: Quality and Patient Safety Environment Grades by State

The category of quality and patient safety environment pertains the most to ambulance operation and equipment. As technology advances we must be willing to "[examine] how better-quality systems and technologies can help improve care and prevent injuries." This is essential in order to advance the emergency medical field. Upgrades in the ambulance are necessary to make acquisition of equipment and medication faster, easier and expose that equipment to less contamination. Such improvements could dramatically improve the quality level of care given as well as make the ambulance environment safer for patients. As shown in Figure 18, significant advancements are needed in this category so that we as a country are adequately caring for our fellow citizens.

2.3.1 Equipment

One of the most important aspects of emergency medical care is the ability to respond quickly and efficiently to patients in distress. Ambulances are the primary vehicle which is capable of providing immediate care and transporting patients to the hospital where they can receive further and more advanced treatment. All ambulances, regardless of state, must be able to provide at least basic life support which means that they must carry devices to assist breathing, immobilization, defibrillation, bleeding, infection control, obstetrics, communication and injury prevention. Medical service vehicles in the United States are required to carry certain equipment as well as medications in order to provide this basic level of support for patients. Ambulances which are cleared to give advanced life support (ALS) must carry additional equipment to deal with patients in critical or severe condition. The amount of equipment is regulated by the state and federal government and can be massive. In some situations, ambulances may be carrying a minimum weight of upwards of 800lbs[22]. Figure 19 depicts an ordinary system of storing medical equipment within an ambulance.

Figure 19: Storage of Equipment within an Ambulance

Ambulances must carry all of the necessary equipment and still allow EMS to be able to find what they need quickly without having to search through a mess of cluttered items. To ensure this, KKK-A-1822F, a federal law, mandates that ambulances have a minimum of 35

cubic feet of enclosed cabinetry, compartments and shelves. Furthermore, to prevent contamination and facilitate cleaning KKK-A-1822F mandates that all patient compartments, including storage compartments, be "impervious to soap, water and disinfectants, mildew resistant, fire resistant and easily cleaned/disinfected."[23] Table 11 shows some of the bandaging equipment carried in San Mateo County, California ambulances and the minimum quantity of each item that must be carried[24].

Table 11: Bandaging Equipment Carried on Ambulances

Bandaging and Splinting Equipment	Minimum Quantity
Extremity Traction Splint (model optional)	1
Cervical Collar (Stiff) (sizes to fit all patients > 1 year old)	1 each size
Rigid Splints	2
Cardboard Splint Arm	1
Cardboard Splint Leg	1
Spinal Immobilization Device W/Straps	1
Head Immobilizer	1
Sterile Burn Pack	2
Tape: 1/2" roll 1" roll 2" roll	1 1 1
Arm Boards: Short Long	1 1

This table features only some of the bandaging and splinting equipment carried on the ambulance. The right column also includes the minimum amount of each item that must be present inside. The full listing of equipment carried in a standard Massachusetts ambulance can be found in appendix C.

2.3.2 Drug Classification

EMT's have a wide range of drugs at their disposal to provide emergency treatment to patients. Choosing which drug to use depends entirely on the condition of the patient and the symptoms they are showing. All ambulances have a need to carry medication and which medications are on approved lists vary from state to state. Table 12 shows an excerpt of the list for San Mateo County, California ambulances and the minimum quantity that must be carried[24]. The full listing of medication carried on these ambulances can be found in Appendix B.

Table 12: Medication Carried on Ambulance

Medication	Minimum Quantity	Standard Quantity
Glucagon 1 mg/vial	1 mg	2 mg
Glucola 10 oz. or Glucose Paste, tube	2	2
Lasix 40 mg	120 mg	160 mg
Lidocaine 100 mg/5 ml preload	400 mg	800 mg
Narcan 2.0 mg/2 ml preload	2.0 mg	8.0 mg
Nitroglycerine 0.4 mg tab or Nitroglycerine metered spray	2.4 mg 2 bottles	 3 bottles
Infant Sodium Bicarbonate 5 mEq/10 ml preload	10 mEq	20 mEq
Sodium Bicarbonate 50 mEq/50 ml preload	100 mEq	200 mEq
Morphine Sulfate 10 mg/1 ml ampule w/ tubex carpuject	20 mg	50 mg
Versed 2mg / 2cc vial, ampule, or preload	4 mg	8 mg
Medication Labels	10	10
Portable Medication Box	1	1

Certain pharmaceuticals are classified as narcotics, which are also controlled substances as defined by the Controlled Substances Act, and therefore cannot be stored so that passengers could easily access them[27]. To prevent unlawful distribution, narcotics are kept in a separate cabinet or storage container that can only be opened with a key held by EMS personnel. Non-narcotic drugs have no such stipulations and therefore may be held in normal unlocked storage cabinets.

The different drugs carried on ambulances have been classified into 5 distinct groups based on both function and method of administration. These groups consist of cardiovascular medication, cardiopulmonary medication, analgesics (painkillers) and anesthetics (sedatives), anti-inflammatory and immunosuppressive drugs and finally miscellaneous drugs. Figure 20 presents each of the five categories as well as the percentage of the total amount of drugs placed in those categories.

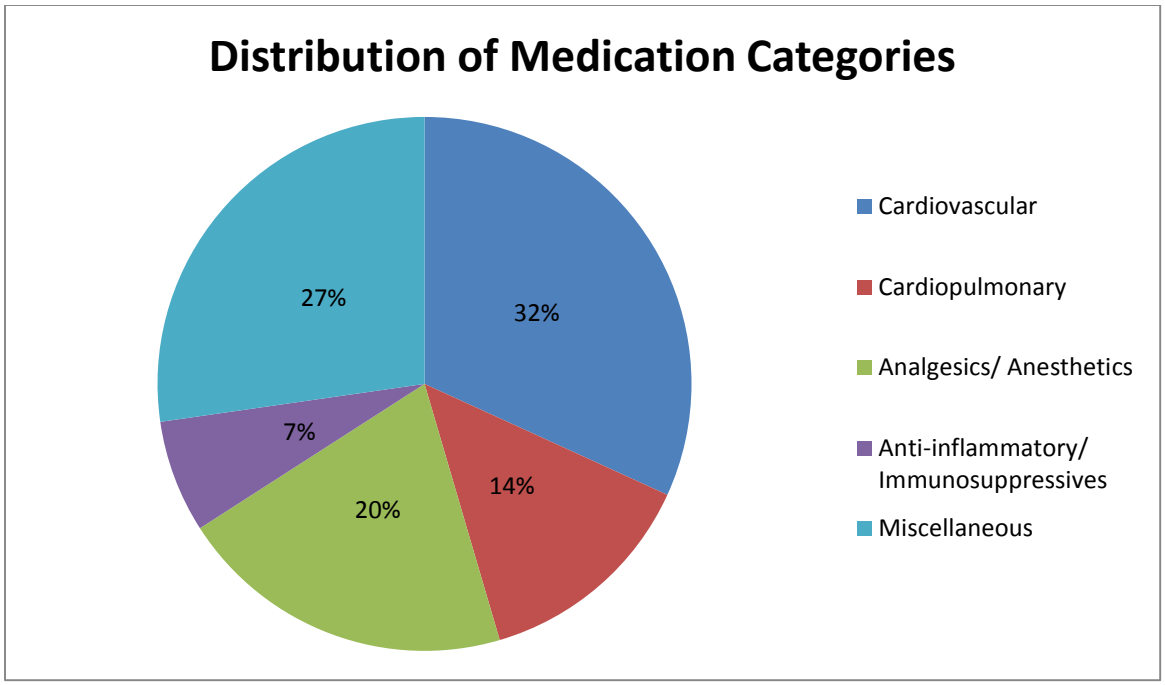

Figure 20: Placement of Medications into Distinct Categories

As shown in the figure, nearly one third of the carried pharmaceuticals are for dealing with cardiovascular problems. The cardiopulmonary and analgesics/anesthetics categories are comprised of one fifth and one seventh of the drugs respectively and a mere one fourteenth of the total drugs are in the anti-inflammatory/immunosuppressives category. Lastly the miscellaneous category contains a little more than one fourth of the drugs.

Table 13 shows some examples of medications carried in some ambulances and their inclusion in one of the five categories. For a more complete breakdown of the medications into groups as well as showing the primary function and method of administration see appendix C. Table 13 shows how the drugs are distributed into groups based on their more general site of action or their effect on the body shown in column two. Column three expands upon this by listing the primary function of treatment of each medication. The method(s) of administration for every medication are given in column four.

Table 13: Selected Grouping of Medications

Drug	Group	Primary Function	Method of Administration
Adenosine	Cardiovascular	Antiarrhythmic Agent	IV
Procainamide	Cardiovascular	Antiarrhythmic Agent	IV
Atropine	Cardiovascular	Anticholinergic Agent	IV
Nitroglycerin Tab or Spray	Cardiovascular	Vasodilator	Sublingual or IV
Diltiazem	Cardiovascular	Treats High Blood Pressure (Calcium Channel Blocker)	Oral
Enalapril	Cardiovascular	Treats High Blood Pressure (ACE Inhibitor)	IV
Aspirin	Cardiovascular	Analgesic, Anti-Inflammatory	Oral
Albuterol	Cardiopulmonary	Bronchodilator	Nebulized Inhalation
Ipratropium Bromide	Cardiopulmonary	Anticholinergic Agent	Inhalation
Tetracaine	Analgesics/ Anesthetics	Local Anesthetic	Topical
Midazolam	Analgesics/ Anesthetics	Antiepileptic	IM
Methylprednisolone	Anti-inflammatory/ Immunosuppressives	Anti-Inflammatory	IV
Hydrocortisone Sodium Succinate	Anti-inflammatory/ Immunosuppressives	Immunosuppresive Drug	IV
Narcan (Naloxone)	Miscellaneous	Opioid Antagonist	IV
Ondansetron	Miscellaneous	Antiemetic	Oral or Rectal

Some medications may have different methods of administration due to different packaging of the drug. For example Nitroglycerin may come in either tablet or liquid form and therefore will have to be administered orally or through IV respectively. When applicable, each group is additionally categorized into sub-groups based on their specific functions. For example Adenosine and Procainamide are both cardiovascular medications and their primary function is to act as antiarrhythmic agents. Method of administration was used as a determining factor because while two drugs may have similar effects, if one drug can be inhaled while another cannot, it will have an effect directly at the lungs while the other one will not. This distinction is noticed only in the case of Atropine versus Ipratropium Bromide which are both anticholinergic

agents yet they are separated into cardiovascular and cardiopulmonary medications respectively.

2.4 History of Vending Machines

Vending machines are able to store and distribute large quantities of items very efficiently thereby saving space. Using a unique code to choose from a variety of pre-organized vended items, it is simple for a user to easily access only what is needed at that time and nothing else. By keeping items separated physically from outside contaminants they are able to ensure safety and efficacy. Vending machines are, at their most basic, a mechanism capable of distributing items in response to some stimulus. This stimulus is usually in the form of insertion of some predetermined amount of money and the depression of a button to designate the item of purchase. Distribution methods vary, but will almost always incorporate some mechanical element and may also, especially in the case of many modern designs, have electrical elements as well.

Vending machines offer a different way of purchasing goods, one devoid of personal interaction but one in which goods are delivered quickly, efficiently and without the added effort of hunting down a specific retail store. Even now, vending machines can be found in almost any location imaginable be it an airport, cafeteria, mall or any other public location. With approximately one vending machine for every 23 people, Japan currently has the highest number of vending machines per capita and is at the forefront of vending machine technology[29]. While the most common items sold are snack foods and beverages (Figure 21), vending machines are now used to market a vast range of items including alcohol, cigarettes, and DVDs. With such a plethora of items now being offered, vending machines may well be the way of the future.

Figure 21: Vending Machines Line a Japanese Rest Stop

In America today, there are about 97 different companies which are responsible for the production and manufacturing of vending machines and vending machine parts. The leader of vending machine sales in the United States is Dixie-Narco inc. which provides major beverage companies such as Coca-Cola and Pepsi with beverage dispensing machines. There are two primary organizations which advocate for vending machine companies; The National Automatic Merchandising Association (NAMA) and The National Bulk Vendors Association (NBVA). Founded in 1936, NAMA is responsible for representing all parties that have ties to automated vending machines from manufacturers to the suppliers of goods that are distributed in the machines. They also collect a broad range of data and even put out periodicals that inform members of relevant state legislation, labor issues and an industry newsletter. NBVA, founded in 1949, differs from NAMA in that they are only concerned with the manufacture and operation of vending equipment[30].

2.4.1 Old Vending Machines

Vending machines have been around for at least two millennia with the first known example of a vending machine dating back to the first century C.E. in the Egyptian temple in Alexandria (Figure 22). This primitive example was used to distribute holy water in which a coin placed in the machine would hit a lever, opening a container and allowing the water to flow[31].

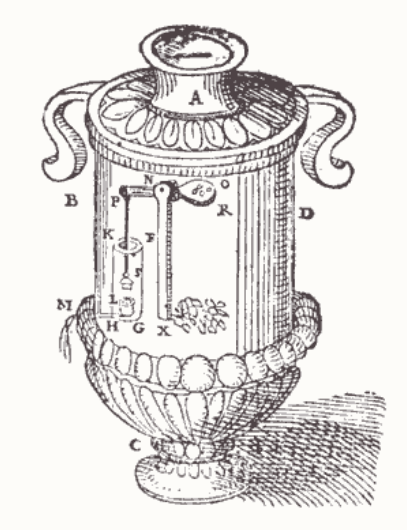

Figure 22: Possibly the Oldest Vending Machine ever Invented

While not many vending machines have operated since this time, they did see a resurgence in popularity that coincided with the industrial revolution. During this time, as population became denser and denser within cities, vending machines started to appear which sold such things as postcards, gum or books. Vending machines could also be found early on at railroad stations and other locations with large numbers of passersby (Figure 23).

Figure 23: Cigarette Vending Machine

These vending machines resemble those that are present in society today but obviously were limited to dispensing products through a strictly mechanical pathway. Although the first patent for a vending machine was awarded in 1857, it was not until the 1880s when many patents were granted for different types of coin acceptor mechanisms that the vending machine industry was truly able to take off. These coin acceptor mechanisms could differentiate real coins from fake ones, which greatly reduced profit loss due to theft[32]. The first vending machine appeared in the United States in 1888 and was built by the Thomas Adams Gum Company. Even after the coin acceptor mechanisms came on to the scene problems still remained for automated vending machines. Problems such as potentially damaged merchandise, customers having access to more than they have paid for and finding other ways of cheating the coin acceptors persisted. These problems would continue for the vending machine industry and are even present today.

2.4.2 Modern Innovations

The first occurrence of electronics being used in vending machines was in the 1970's with the "talking" machine. Following a purchase, the vending machine would play a recording of a one-liner by a famous comedian. However it was not until the 1980's that electronics were incorporated into the distribution mechanisms of vending machines. In the 1990's "smart" technology came to dominate the industry. With this technology, vending machines can keep

records of their inventory and alert the company when they need to be restocked, diagnose glitches or report damage or unauthorized entry into the machine[30].

A commonly found snack-dispensing machine is shown in figure 24. The mechanism for distributing items is to store snacks on each rung of a corkscrew with the end of the corkscrew leading to a drop-off point. This setup ensures that when the corkscrew undergoes one revolution, each item on it will advance one rung towards the drop-off point. Whenever an item gets to the drop-off point it will fall a short distance into the collection area from which it can be picked up by the costumer. A letter pad or a number pad (or sometimes both) are used to determine the snack that the costumer wants; with each snack corresponding to a different letter-number combination. When the correct combination is pressed and the correct amount of currency is inserted, the machine will electronically send a signal to spin the chosen corkscrew one revolution which in turn forces a snack to fall into the collection area.

Figure 24: A Typical Snack Vending Machine

Many vending machines now use a touchscreen surface to display their goods. Touchscreens allow items to be chosen directly by clicking on the item shown on the screen. In this way, the screen makes it difficult for someone to enter accidently the wrong combination, and therefore receives the wrong product. The use of a touchscreen can also allow for more items to be stocked within the machine because the vending machine manufacturers do not need to worry about separating items in a way that makes each item distinguishable visually.

Telemetry is among the newest innovations in the vending industry. With telemetry, vending machines can send data such as what items were purchased, and which location it was purchased at. Some companies such as Intel are aiming to collect even more advanced data regarding approximate age and sex of the costumer, how many people viewed the machine, and even what items they were looking at[33]. By using telemetry, retailers can get a better picture of what to stock their machines with, when to restock them, who they are marketing to, and even manage to optimize the placement of vending machines within a building or marketplace to get better rates of sales. While much progress has been made in the industry, there is still plenty of room left for improvement and further innovation.

CHAPTER 3. Automated Dispensing Machine

This project seeks to replace the current storage system of cabinets with a vending machine capable of holding and dispensing pharmaceuticals and consumables. The proposed vending machine is designed to be placed on the left wall of the ambulance and will run from the rear of the ambulance to within reach of the captain's chair. On the side of the vending machine nearest the EMT there is both an interface for the EMT to interact with the vending machine and a retrieval site where the EMT can get the pharmaceuticals that were requested. The vending machine model was designed to hold items in corkscrew holders which, when rotated by motors, force items to fall onto a conveyor belt. For medications to be held properly in the holders, drugs must be prepackaged before being stored in the vending machine. After falling, the conveyor belt then takes the drugs to a retrieval site close to the captain's chair where an EMT may pick them up. Items may be requested from the vending machine by using a touchscreen interface. The interface is just one aspect of the control system for the vending machine. The control system is made up of the electric system, a microcontroller, and the user interface. The interface communicates with the vending machine via the Arduino Uno microcontroller which is capable of controlling the electrical system in order to run the corkscrew dispensers and the conveyor belt. The control system thereby allows EMT's to request and retrieve pharmaceuticals and consumables from the vending machine.

A series of tests were proposed in order to assess whether or not the vending machine was successful in meeting its original objectives. Retrieval time for items, stresses experienced by EMT's, loose items, and contamination of stored items constitute the proposed tests. Expected results are given for each test and those results are analyzed in order to give a general idea of how the vending machine compares to the current storage system in ambulances. A proposed survey is also shown with example questions to ask EMT's regarding the effectiveness and usability of the vending machine.

3.1 Design

The purpose of the design is to store, protect, retrieve and dispense the pharmaceuticals and consumables. Designing the automated distribution machine presents a unique challenge because of the limited space. The design space is located in the rear compartment across from the ride-along chair by the rear-door. How the design meets these objectives can be easily understood when each individual component is taken into consideration. The housing fits within the design space and is anchored by attaching to the ambulance box. The housing also must create a barrier between the items and the patient compartment so that contamination of medication is prevented. Three different sized dispensers were designed to securely hold different sizes of packaged pharmaceuticals. The items need to be packaged differently so only the required dosages are stored and dispensed. The dispensers deposit the items to the delivery system, and the delivery system must transport the items to the collection area. The collection area must be within arm's reach of the EMT. The final design can securely store items within the dispensers. The items are protected by the housing before being dispensed to the collection area where they can be easily retrieved.

3.1.1 Location in the Ambulance

The Interior storage area of ambulances is limited. The effective use of the workspace in the ambulance compartment is imperative. Because ambulances have to fit on roads and are driven by a car motor, there is a restriction on how much storage space can be created. The proposed storage device cannot be too heavy as it can overburden the car motor, and can cause the ambulance to breakdown. Understanding the spatial constraints was the first step to creating a correct design space to work within. Figure 25 shows a floor plan proposed by the previous design group[22]. There is a space to the left of the patient stretcher for the placement of a new device. This 20 inch by 72 inch by 80 inch area is the design space that we must work within if the new storage device is to fit. One of the concerns that would stop a conventional vending machine design is the lack of depth to work with. A lot of conventional vending machines that are seen around the US have a depth closer to 40 inches. This depth allows for a

large store of items and plenty of room for a delivery system. These vending machines are normally not 80 inches wide. Alterations to the normal vending machine design are clearly needed to fit the desired design space.

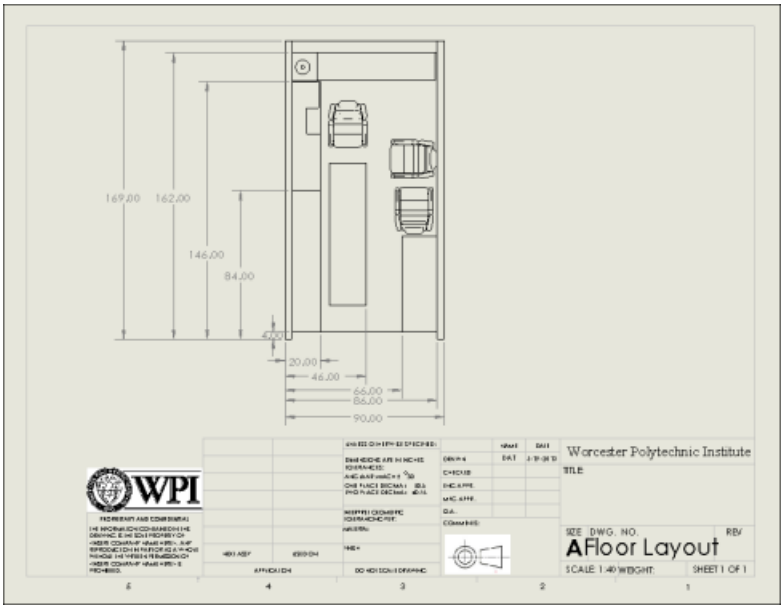

Figure 25: Proposed Floor Plan from a Previous Project

The lack of depth is the most constraining aspect of the design space as depth in a vending machine translates to carrying capacity. To make up for this lack of carrying capacity appropriate dimensions for the design are considered. Figures 26 and 27 show the dimensions of the housing for the vending machine. The total length of the box is 4 inches longer than the proposed space. These extra thick walls would be part of the ambulance box and would not take away from the design space. The small extrusion in the bottom right of the housing is the collection area. It is closed off on three sides to limit the access to the dispensed items. The side facing the captain's chair is left open to allow the EMTs to easily reach and access the dispensed items. The back of the captain's chair is 38 inches from the collection area. In Figure 27 the transparent material makes it possible to see the hole at the top of the collection area, which allows items to exit the machine. There are two panels on the front of the housing that allow access to the interior of the machine. The small panel on the bottom allows access to the delivery system and the larger top panel allows access to the stored items. The bottom panel is made of a thin light metal such as aluminum, while the top panel is made of a clear plastic. The clear plastic allows EMTs to see that the item is getting dispensed properly and the general

state of the storage. Both panels can have a locking mechanism that connects to the middle divider to prevent theft while the ambulance is not in use.

The depression in the upper right of the housing is a space for a computer system that would control the vending machine and a display that allows EMTs to easily control the system. Originally there was a space below this depression for a number and key pad, but changes in the programing made a touch screen interface more viable. This removed the need for a keypad interface and created space for additional changes to be made later.

The final aspect of the housing is the back panel which can be clearly seen in Figure 27. This back panel should be made of the same lightweight aluminum or steel as the housing and separates the electronics and the storage. During manufacturing, holes can be punched out of the back panel to allow the connection of the dispensers to the motors. Because there is a back panel separating the electronics, any maintenance would have to be done from the outside through an access panel along the back of the housing.

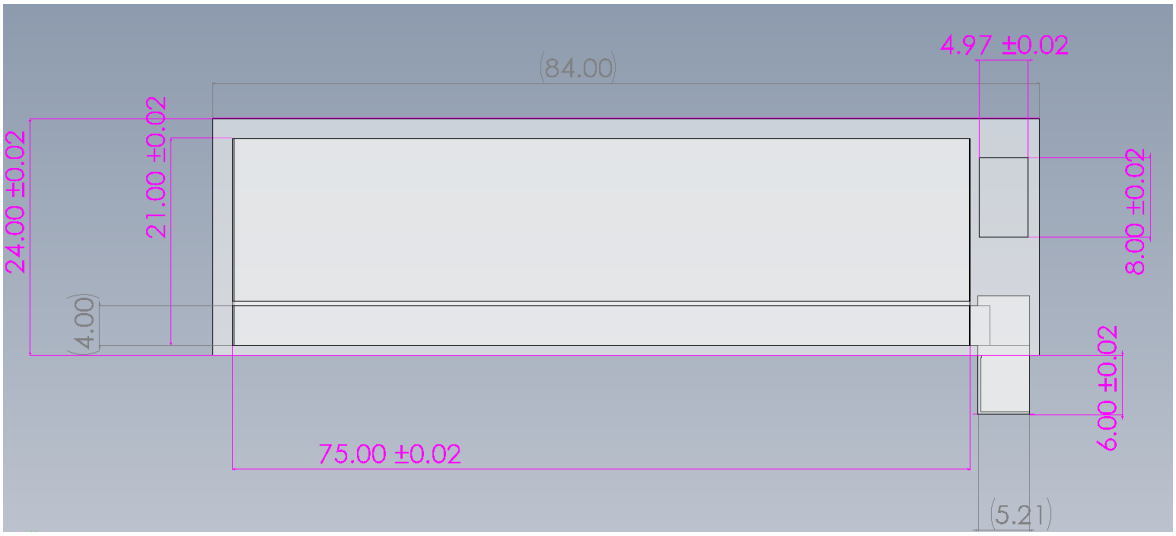

Figure 26: Front view of Housing

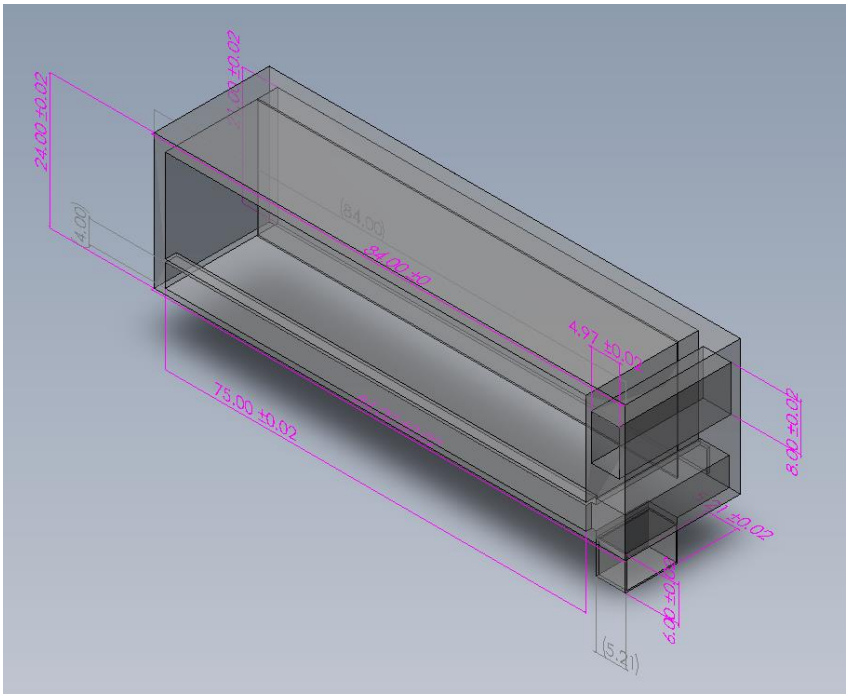

Figure 27: Isometric View of the Housing of the Vending Machine

3.1.2 Dispenser

The dispensers in many current vending machines fall into one of two categories, a tabs dispenser or a corkscrew dispenser. The tabs dispenser works by spring loading the items to be pushed forward. Two tabs at the end of the dispenser stop the items from falling, and when an item is requested the tabs retract and allow an item to fall. They then move back into place and stop any further items from being dispensed. This method works for large solid items like bottles, but for many items found in the ambulance this would not work. If the item deforms it is difficult for the tabs to stop it from moving forward. It is also difficult to stop just one item form being pushed out of storage, which would waste the second unused item. Programming the outputs for this kind of system would also be difficult. In soda vending machines it works well because the items all have the same shape and respond to the spring force similarly. This is not the case for the ambulances as the items are varied in size and physical properties. The tabs would have to retracts and extend differently for each item, which would require a different output for each of the items in the program. It would also be difficult to calculate the exact time need for only one item to be dispensed.

The second option is a corkscrew dispenser, which used a large spiral to hold and move the items. A motor generates the torque that rotates the spiral. When the spiral rotates the items between the coils are pushed further up the spiral. Eventually the item gets to the end of the spiral and fall forward. This is more ideal because the motors can be used to rotate the spiral one revolution per item request. This stops the issue of having more than one item dispensed at a time. The problem that this system runs into is if the item does not fall when it reaches the end of the spiral. This would be worse than if two items fell when only one was requested. To solve this, the spiral does not end where the item drops. There is a hole in the tube that allows the items to fall before the spiral ends. The spiral will rotate so that the item has to be completely over the hole. This is a problem is if the items are not relying on the tube structure to support their weight. Another challenge that comes with this design is that nothing can obstruct the items downward path. This means that the dispensers have to be terraced to allow items to fall unobstructed to the delivery system. Terracing the dispensers also cuts down on the number of items that can be stored in each subsequent layer of dispensers.

To accommodate for a range of item sizes three differently sized dispensers is recommended. The three different sizes can be seen in Figures 28-30. The diameters of these tubes are 3.5 inches, 2.5 inches and 1.5 inches. The space between the spirals ranges from two-thirds of an inch to one-half an inch. Up to seven or eight items can be stored at a time in each of the dispensers. A slider attached to the end of the spiral which is connected to the driveshaft of a motor. This slider ensures that the spiral stays aligned with the tube surrounding it. The slider should fit perfectly into the end of the tube with a small groove for it to be set in or to be connected to the tube with ball bearings. The tube, slider and spiral should all be made from a plastic to reduce cost and weight.

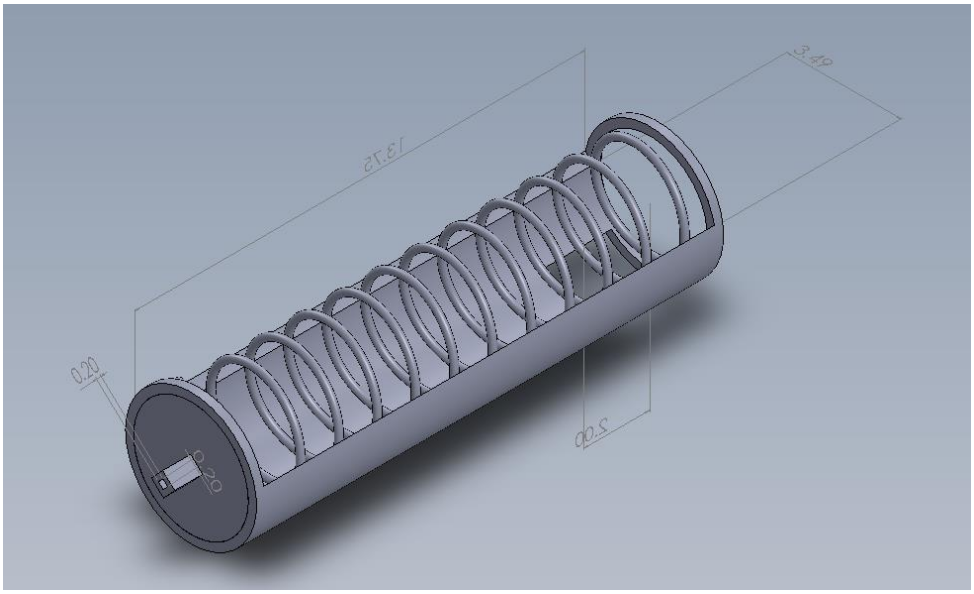

Figure 28: Isometric view of Size 1 Dispenser

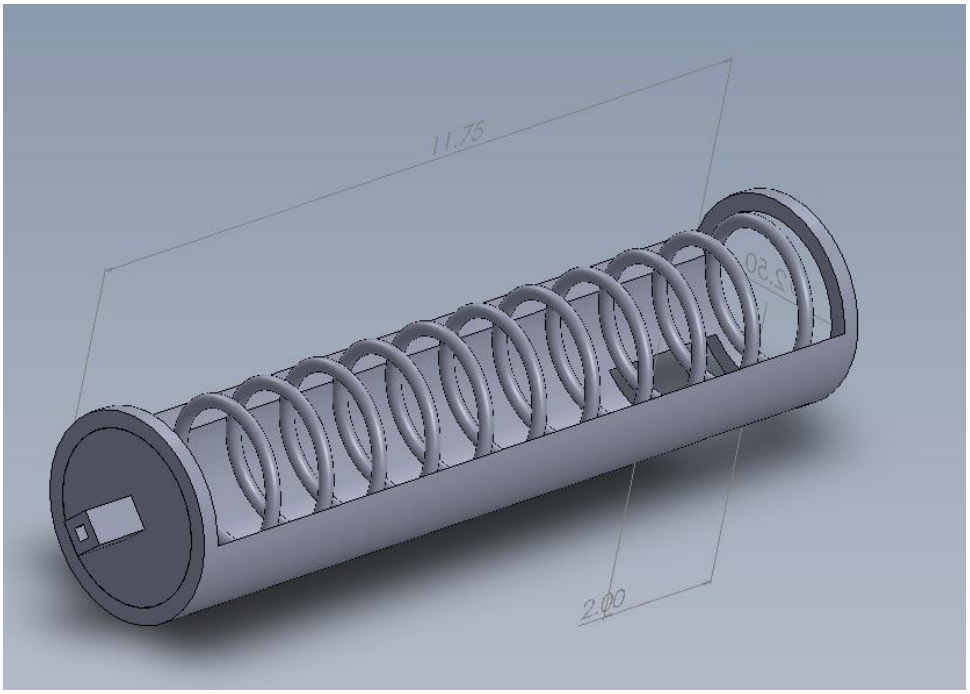

Figure 29: Isometric view of Size 2 Dispenser

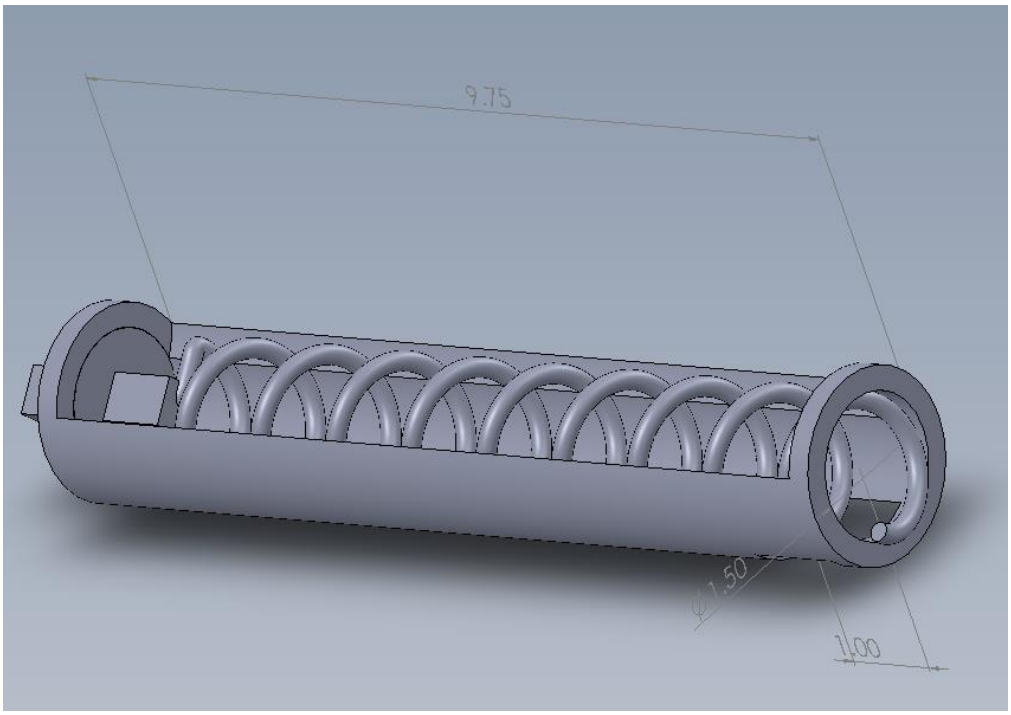

Figure 30: Isometric View of Size 3 Dispenser

3.1.3 Packaging of Medication

In order for the vending machine to meet its intended requirement of storing and dispensing pharmaceuticals, modifications to the current packaging must be made. The best option of storing drugs inside the vending machine would be to prepackage each type of drug and stock those packages within the machine. A drug is considered to be prepackaged if it has been "removed from the original manufacturer container and placed in a dispensing container for other than immediate dispensing to a patient."[35] Drugs which are prepackaged require labeling which contains the name and strength of the drug as well as the manufacturer's name, expiration date and lot number. This is necessary so that drugs will not become mixed up and therefore it will prevent administration of the incorrect drugs to patients in emergency situations. Although consumables must also be prepackaged, due to the great discrepancies in shape and size it would be difficult to individually address each item in this report. Because of this, only the packaging of pharmaceuticals will be considered.

Drugs which come in solid form whether in pills, tablets, capsules or gels are often kept in what are known blister packs. Blister packs, shown in Figure 31, utilize a plastic coating on

one side which is shaped in such a way to accommodate the drug. On the other side is a sheet of paperboard coated in aluminum foil which, along with the plastic, encapsulates the drug. Accessing the drug is as simple as pushing on the plastic casing hard enough so that the paperboard plane breaks. The drug will then emerge from the other side ready to be administered to the patient in the proper dosage.

Figure 31: Pills Sealed within a Blister Pack

Blister packs are very useful for protecting drugs from their environment. Since the drugs are perfectly sealed within the cavity they are not affected by changes in humidity nor are they susceptible to contamination until opened. This leads to sustained efficacy over longer durations of time. A potential disadvantage of using blister packs is that if the user is not careful, the pill can sometimes pop out uncontrollably, and it end up on the floor or being contaminated. In an ambulance environment it would be unethical to administer unprotected drug to a patient. Thus a new pack must be opened. However this would be unlikely if the EMT were being careful and paying sufficient attention to opening the package. For the blister pack to be usable in the vending machine holding system, it must fit into the small holders. For it to do so, the pack must be shorter than 3 inches in both height and length and less than 0.5 inches in depth. Figure 32 shows the proper 40mg dosage of the Lasix (Furosemide) medication

packaged in its current blister pack and the dimensions that the total package would assume. The dimensions of the newer packaging for the vending machine will be 2.5 inches long and wide. The depth of the shown package will be negligible at all areas not containing the pharmaceutical but will be about 0.1 inches in the center of the blister pack.

Figure 32: Blister Packaging for 40mg of Lasix (Furosemide)

The majority of the depth requirement of the packaging is taken up by the actual medication itself and pill or tablet form drugs are rarely thicker than 0.25 inches. The packaging for the drug shown in Figure 32 is approximately 0.1 inches deep. For any drug, the packaging would only add, on average, about 0.05 inches of depth to each drug, still allowing plenty of leeway in the size of the drug while still fitting into the machine holders. Normal blister packs are not very large and usually take up only about 1/2 to 1 inch in length and width. Because the space within the holders is too large for normal blister packs to make good enough contact with the corkscrew, blister packs with a larger area must be used. Since the blister pack shown in the figure has both a height and a width of 2.5 inches, it can fit more snugly into the dispenser. The dispenser will therefore be better able to move the drug forward as it rotates. It should be noted that the pill itself is about 0.4 inches in diameter and is under 0.1 inches deep so the

packaging shown is not proportional to that which would be used to store the drugs in the vending machine.

Liquid pharmaceuticals or medications which must be delivered in solution require a different means of packaging. One commonly used option for packaging liquids is to have the drug stored in prefilled syringes. Prefilled syringes, shown in Figure 33, come with the correctly measured amount of drug stored in the barrel and the volume of the drug can be verified by the EMT by reading the measurement lines on the side of the syringe. The needle is covered by a protective cap which is usually secured to the hub of the syringe by screw threads or simply by friction. Once the cap is removed, the drug may be administered to a patient by depressing the plunger.

Figure 33: (Left to Right) 3, 5 and 10mL Prefilled Syringes

There are many advantages to using prefilled syringes as a way to package liquid medication for intravenous or intramuscular injection. In some cases a small difference in drug delivered can have a huge effect on the patient so it is imperative that the correct amount of drug be administered. Prefilled syringes not only come with the proper dosage inside but also have volumetric measurements on the side so that the EMT administering the drug can verify

that he or she is actually giving the proper dosage of drug to a patient. These preventative measures ensure that the patient is not given an overdose or not enough of the necessary drug. The protective cap is another important safety feature because it prevents unintentional loss of the drug by disallowing depression of the plunger. Not only does this prevent loss of the drug, it also prevents contamination from getting in through the needle and thereby maintains sterility of the needle. Lastly having the needle covered is important because it prevents accidental injection. A main reason to use prefilled syringes however is their ability to save large amounts of time. Without a prefilled syringe the EMT would have to fill his or her own syringe with the proper amount of medication. This requires the EMT to gather an empty syringe, the medication and then accurately and precisely uptake the drug all while in a moving vehicle- an act which is not always easy to perform.

Prefilled syringes would be stored in the large dispensers which means that they must be no taller than 5 inches (although between 3 and 4 would be preferable) and be no wider than 3/4 of an inch. Figure 34 shows an empty and fully depressed 10 mL syringe to be stored in the vending machine. The barrel of the syringe will be 2.2 ± 0.3 inches long as will be the plunger and the needle will be approximately 1 inch in length. This gives the entire syringe a length of about 5 inches when fully elongated due to a 0.5 inch overlap between the barrel and the plunger. The barrel of the syringe will have a diameter of 0.6 inches and the extruding part at the base of the barrel will be 0.1 inches long on each side. Therefore the widest area of the syringe will be at the base of the barrel and is about 0.8 inches wide.

Figure 34: A 10mL Disposable Syringe

The figure has a depth of 0.8 inch: slightly larger than the 3/4 inch limit for the holders. This does not present a problem because when the syringe is placed in the holders the extruding part can lay parallel to the length of the dispenser. In this orientation, the part of the syringe fits through the open space between the corkscrew. In this way, even if the syringe exceeds 3/4 inch in width it can still be accommodated in the holder and will function normally. The other problem presented by the syringe is its large height. The syringe has a height of 5 inches which is nearly the limit of what can be held in the dispensers.

The most commonly used inhaler is the metered dose inhaler or MDI which administers preset amounts of the drug when the inhaler is used. Modifications to the MDI have given us the disposable inhaler, shown in Figure 35, which contains just a single dose of the medication. This disposable inhaler can be modified to fit into the proposed vending machine.

Figure 35: A Disposable Inhaler

For a disposable inhaler to fit inside the large vending machine dispensers it would, like the syringe, have to be less than 5 inches in height and have a width of no more than 3/4 of an inch. The height requirement is not a problem for a disposable inhaler which is between 4 and 5 inches tall. The width requirement however is a major problem which may not be resolved in this project. Inhalers which are currently on the market have a diameter of about 1 inch in the long section but then branches out to include a mouthpiece which contributes an extra inch to the total depth. Figure 36 depicts a standard inhaler which demonstrates this problem.

Figure 36: Side View of an Inhaler

Although the syringe was able to overcome this setback due to its shape and the less prominent size discrepancy between it and the holders, the disposable inhaler would simply be too bulky to fit in them in any orientation. Adding to this problem is the irregular shape of the inhaler. Because it is not a linear object, there is the possibility that instead of falling straight down once it reaches the drop off point, it could instead get caught on the front and rear rungs of the corkscrew.

3.1.4 Delivery System

Once the items leave the dispenser they need to reach the collection area near the EMT. In vending machines this is typically done in two ways. The first is that the item just drops done into the collection area to be retrieved. This would not be useful in the design as the goal was to reduce the amount of movement the EMTs need to get items. The second common approach is a two axis retrieval unit. The unit acts in to separate axis to reach any point in the machine. This allows it to retrieve the item from the dispenser and transport it to the collection area. This method takes a very long time depending on how far the unit must go to retrieve the item.

To simplify the process the design should utilizes a different two axis system. One way to do this would be to have a slide the can bring any item that falls on it to the collection area. If

there was a cheap and light weight material that has almost no friction regardless of what is on it this may work. Since no materials were identified during initial searches the final design will not have a slide as the delivery system, but in the future if such a material becomes available the slide would be preferred. Another reason a slide is less desirable is that it takes up a lot of space. Since space is already the most limiting factor, having unused space would take away from the overall storage capacity. The final design uses a conveyer belt instead of a slide or a two axis retrieval unit.

The delivery system still uses a two axis system it can deposit the items closer to the EMTs. The item is allowed to fall which is the motion in one axis, and a conveyer belt moves it along in the other axis. The proposed conveyer belt, shown in Figure 37, is long enough to bring all items to the collection area and small enough that it does not waste a lot of space. The belt of the conveyer should have some sort of tread to increase the friction between it and the items on it. If the items do not gain traction of the belt than the will roll and not reach the EMT in a timely manner.

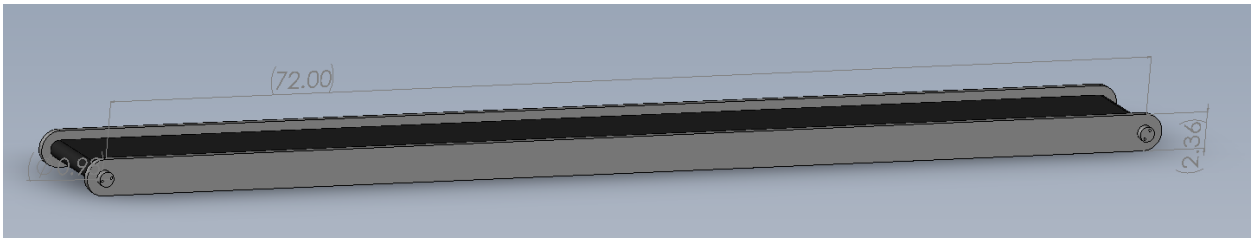

Figure 37: Isometric view of Conveyer belt

3.1.5 Vending Machine

The final model combines all of these components into one assemble seen in Figures 38-40. The dispensers are lined up by size. The largest dispensers are at the top and the smallest at the bottom. They are spaced a quarter of an inch apart to allow for supporting structures to be placed. The supporting structures should be a light metal such as aluminum. There should be two of these supporting structures to hold the dispenser in place. One of the structures should be placed towards the back, which is represented by the plate seen in Figure 40, which stops the dispenser tube from turning with the drive shaft of the motor. The front structure should support the tubing on three sides to prevent unwanted movement.

There are seventeen large dispensers, twenty-two medium dispensers, and thirty small dispensers within this model. There is also enough extra vertical space to place another row of dispensers to increase item storage. This space could be used to rearrange the dispensers by grouping instead of by size. Rearranging it by grouping would make stocking the vending machine more logical. To restock the vending machine it is recommended that extra dispensers are used. Since the dispensers are identical within a size one needs only remove the empty dispenser and place a stocked one in its place. This allows for restocking outside of the vending machine where there is more room and access.

The motors would all be aligned against the back wall and firmly secured. Making the part of the back that holds the motors removable would make fixing anything that break easy. There should also be a form of ventilation that protects the vending machine from overheating and freezing. In the event that the machine should break during use the front panel protecting the items can be lifted up and the items accessible that way. Another option would be to keep some form of the previous cabinet design in the space under the vending machine. The cabinet space could be used to store spare pharmaceuticals and consumables as well as non-vendible items.

Depending on the computer system used for the interface more space could be created within the housing. If a tablet is used for the interface and display of the vending machine then there would be room for a few more dispensers. Those dispensers would be placed in the area right over the collection box. This area could also be used to create a protective flap that would prevent contamination from entering the main housing. Adding a flap to the collection area is a possible future improvement that could allow the EMT to protect items requested by accident. By making part of the housing above the collection area transparent with a clear plastic window the EMT could see which item or items were dispensed. If the flap prevents items from falling into the collection area could be lowered to drop the items and raised to deposit the item in a small storage space. That small storage space would be located towards the wall by the collection area and another access panel could be built into the housing. Unwanted dispensed items could be stored, removed and restocked as they remain free of contamination. With

further research and testing more item specific dispensers might save space and improve functionality and reliability.

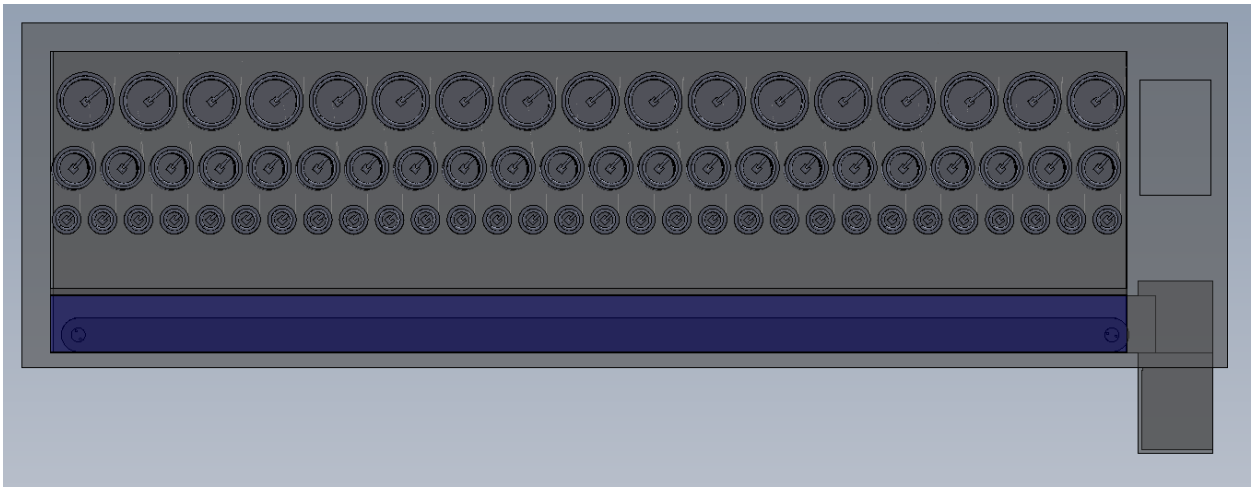

Figure 38: Front View of Final Assembly

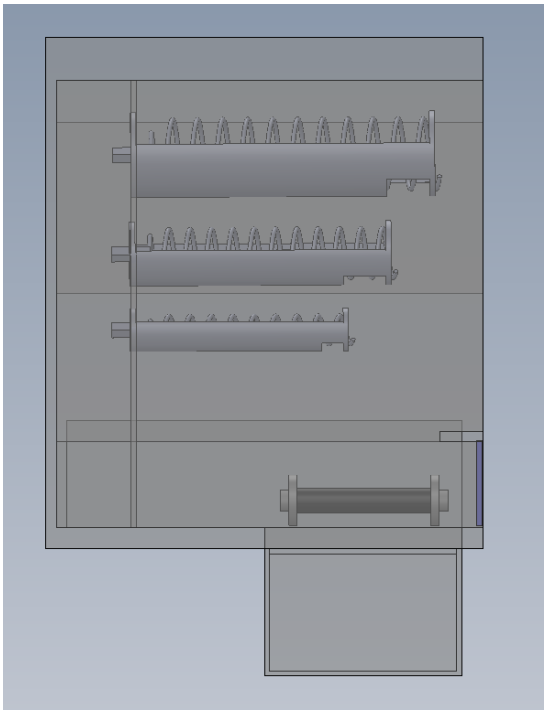

Figure 39: Side View of Final Assembly

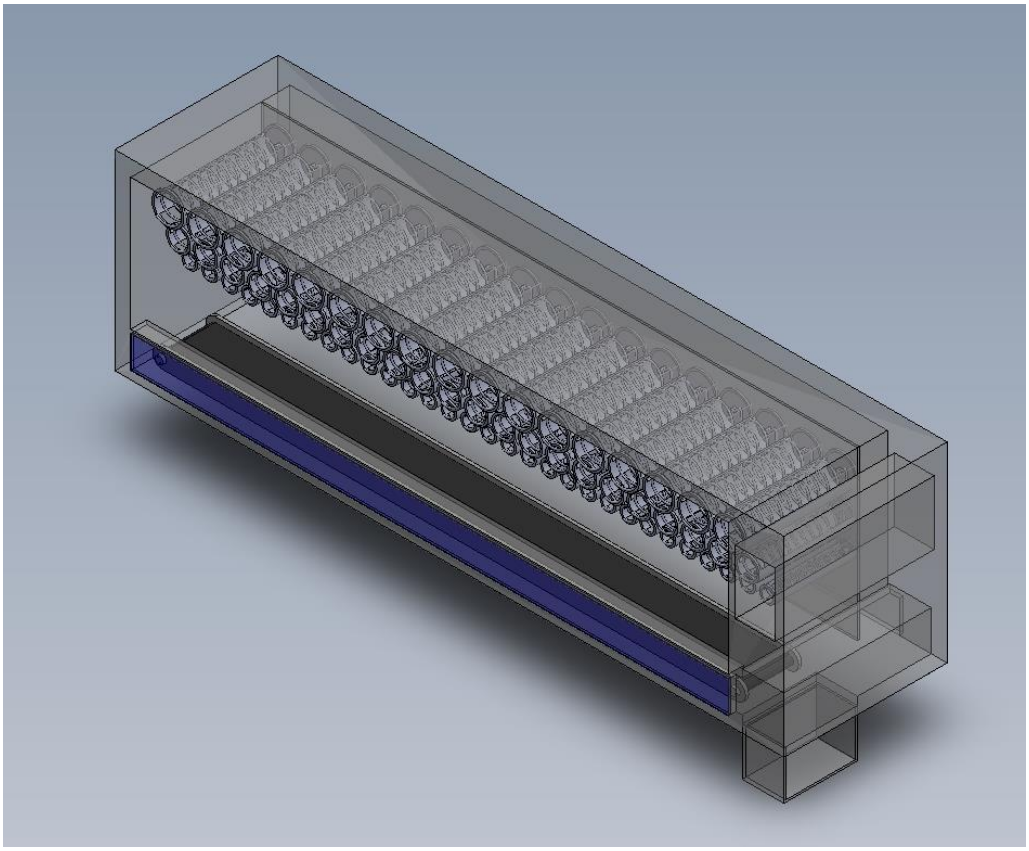

Figure 40: Isometric View of Final Assembly

3.2 Control of the Vending Machine

In order for our vending machine to function it requires a control system. The three key parts of the control system are the motors and motor drivers, the Arduino microcontroller, and the computer program providing the user interface. Figure 41 shows how these components interact.

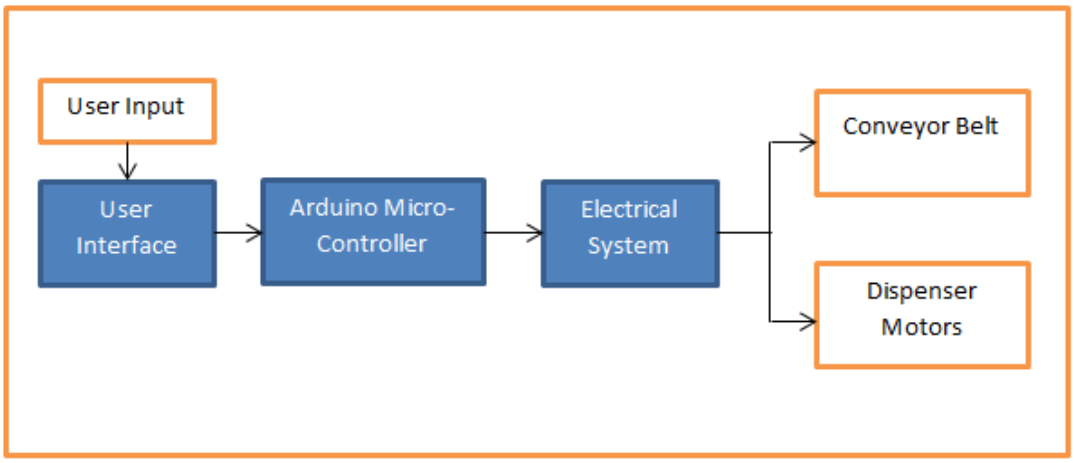

Figure 41: Diagram of the Control System

In the Figure, the elements of the control system are shown in the blue boxes while the inputs and outputs are shown in the orange boxes. These components convert the user input into actions performed by the vending machine. First the computer program converts the user's selection into a signal that can be interpreted by the Arduino. Then the Arduino controls the actions of the vending machine based on the signal provided by the computer. Lastly the motors and motor drivers take the signals from the Arduino and perform the actions in the vending machine.

3.2.1 Electrical System

The electrical system of the vending machine takes the signals from the micro controller and controls the vending machine. To provide this functionality there are two types of components that must be controlled. The first component being the conveyor belt and the second component being the dispensers that hold the pharmaceuticals. We explored using both motors and servomotors to run the vending machine. We considered servomotors because they use less power and can be driven directly from a signal from the Arduino and

determined that the limitations were the power and speed of the servomotors. We considered DC motors because they can supply more power and speed but have the disadvantage of using more power and require an amplifier circuit as the output of the Arduino cannot provide the power to drive the motor. Figure 35 shows how we would wire a relay to boost the signal from the Arduino. In the figure Vsig would be the signal coming from the Arduino that needs to be amplified. Vcc would be the voltage that we want to supply to the motor when the signal from the Arduino is high. Vmotor would be the output of the relay to the motor. The speed of the motor and the capability to move pharmaceuticals is a high priority for an ambulance. Because of this we chose to use a DC motor for the vending machine. The power needed to move the motors needs to be larger than the power the Arduino can generate as output. Therefore, in order for the Arduino to control the motors, it must be connected to a relay. This will allow the Arduino to provide enough power to run the motors.

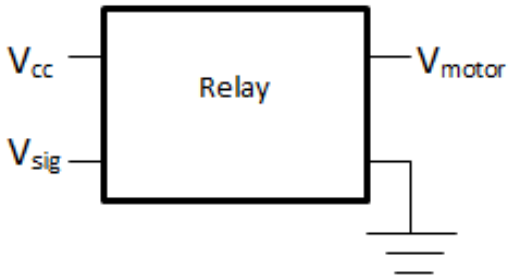

Figure 42: Relay Wiring Diagram

The challenge with having a powered vending machine in an ambulance is that the electrical system is overtaxed as it is now. To limit the power dependent of the vending machine on the ambulance the vending machine needs to store its own power. To achieve this, the space behind the conveyor belt has enough space for batteries that can provide power to the vending machine.

3.2.2 Arduino Program

The first step we took for programming was to create a program that runs on an Arduino Uno. The use of this program is both to provide a proof of concept for the vending

machine controller and a way to test the data sent from the main program running on a computer. Figures 43 and 44 show pictures of what the Arduino Uno looks like. We are interested in pins 0 – 13 which are shown on the top of the board and the USB port that is in the upper left of the board. Of the 14 pins we cannot use pins 0 and 1 as they are shared with the USB port. If the motors are controlled using a constant signal, the pins 2 through 13 can be utilized. On the other hand, if the motors are controlled using Pulse-Width Modulation (PWM) then only the pins 2 through 7 can be used. A useful feature of the Arduino Uno is being able to attach an internal timer to an interrupt. An interrupt is an event that suspends the execution of the main program and runs the interrupt code. After the interrupt code is finished the main program resumes. By attaching a timer to an interrupt the interrupt code can be run repeatedly at regular intervals.

Figure 43: Arduino Uno Microcontroller

Figure 44: Close-Up of Arduino Uno Output Pins

The functionality provided by the Arduino program allows us to send data serially using a USB port. Figure 45 shows the program as a flow chart to illustrate what is happening in the Arduino. The way the code is currently written the data flow is limited to receiving one byte every tenth of a second. A byte is sufficient for our needs since it represents a value from 0 to 255 and the vending machine should not need to hold 255 items. The way the program is written the user can assign a byte value to a pin as a constant that the program will associate

with that pin on the Arduino. Currently the way the code is written the output of a pin is given as a binary signal. The program reads in a byte and if it is associated with a pin the Arduino outputs to a pin for a set period of time. If the Arduino is configured to control the conveyor, the Arduino uses pin 2, which is the pin set aside for the conveyor, to output the signal to drive the conveyor and if it is not configured to control the conveyor, pin 2 can be used like other pins to control a dispenser. The option for the Arduino not to control the conveyor is because one Arduino does not have enough outputs to control all the dispensers and more than one Arduino would have to be used. The program is designed to allow certain pre-defined constants to be changed. The constants include the byte value associated with each pin, whether the first pin controls the conveyor, if the pin's output is a PWM signal or a constant signal, and the duration of time which the conveyor and motors run for.

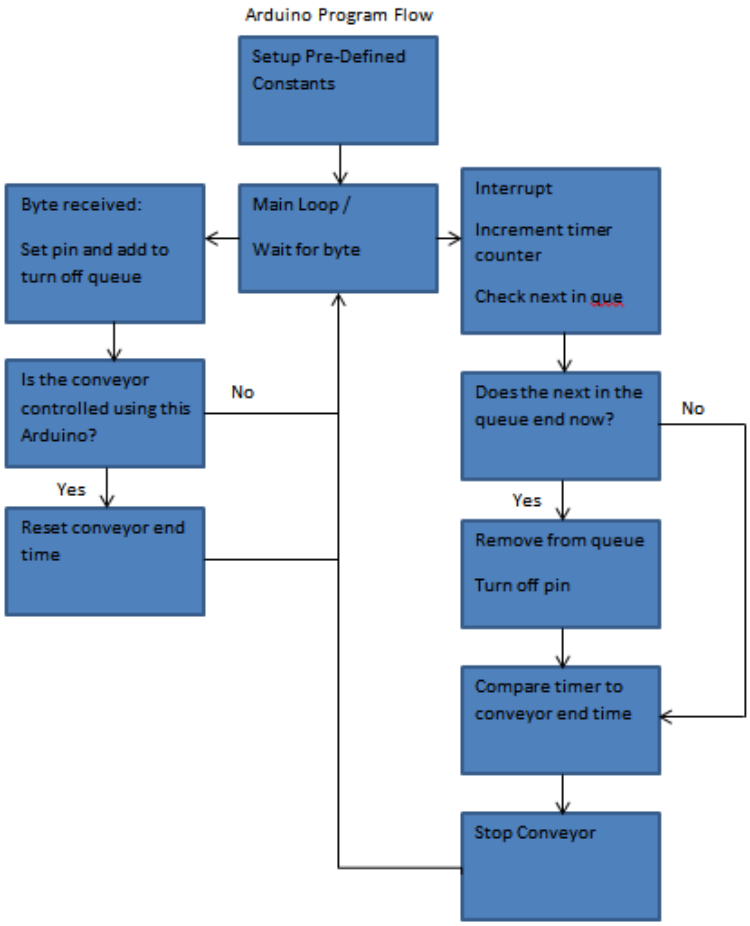

Figure 45: Flowchart of the Program that was written for the Arduino Uno

The Pins 2 through 13 can be assigned to a corresponding byte value. Pin 2 can also be configured to behave as a control for the conveyor or the same function as any other pin. When a non-zero byte is sent, it is checked against the expected value for each pin. If one matches the expected value, the corresponding pin is activated, and the active pin and the time when it should shut off are added to a queue. If the microcontroller is used to control the conveyor after receiving a non-zero byte and before checking for a pin, the conveyor is activated and the time to stop the conveyor is reset to the maximum time. The other event that controls the program is the interrupt which is tied to a 10Hz timer. Once an interrupt has started a counter is incremented. Then this new value is compared to the next entry in the queue. If the counter matches the queue then the associated pin is turned off and the entry is removed from the queue. This is accomplished by cycling the queue so that the second entry is placed as the first entry. After dealing with the pin actions, the interrupt checks to see if the counter's value matches the target value set for stopping the conveyor. If the value of the counter is less than the target value, the conveyor keeps running. If it is equal or greater however, the conveyor is stopped. This code is not limited to the Arduino Uno and can be adapted for most microcontrollers. The only limitation on the microcontroller is that it needs to be able to attach a timer to an interrupt.

3.2.2 Java Program

The design of our vending machine is loosely based on ideas found in common vending machines that use electrical control systems. This differs from the SLATE Ambulance MQP's previously designed vending machine which was entirely mechanical in nature. To allow EMT's to access the contents of the vending machine there must be a way to select the items that they want to retrieve. We created prototype software to test features that are needed to be included in the program. We have attempted to create a user friendly interface that allows the EMT's to easily select one or more of the pharmaceuticals that are stored in the vending machine. The program shows the user what pharmaceuticals are in the vending machine and displays them in a way that is easy to navigate.

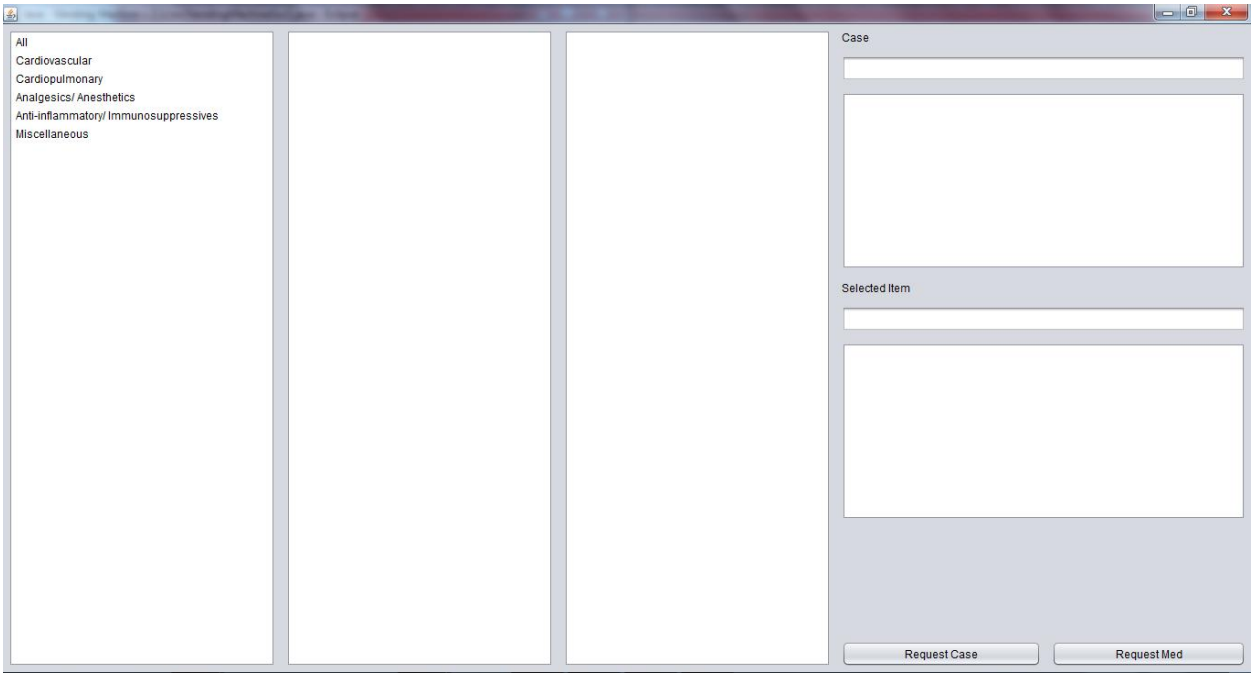

Figure 46: Screen Shot of the User Interface

Figure 46 shows a mockup of the Graphical User Interface (GUI) before any selections are made. The three lists on the left are used for selecting the category, the case or treatment, and the pharmaceutical respectively. The program limits the selection of each list to one item although a future version can have the ability to select multiple categories, cases, and pharmaceuticals. There are two description boxes on the right side of the GUI. One shows a description of the selected case, and the other is used to show a description of a selected pharmaceutical, if either one is applicable. There are two buttons for submitting a request. One button is used to request all of the pharmaceuticals for a selected case while the other is used to request a single pharmaceutical if one is selected. Having two buttons is an issue that we identified as a possible source of problems for the EMT's who would have to remember which button performs which action.

In addition to the user interface, the program also features a logger that writes to a text file that keeps track of the program during key events. This logger is the prelude to a system which can keep track of who is requesting what pharmaceuticals from the vending machine. Another feature of the interface that the EMT's will not be directly exposed to is the use a file system which allows the pharmaceuticals in the vending machine to be changed without having to reprogram the software. The file system has four different kinds of files. There is a

configuration file that is used to define the COM port that the program should connect to and points to the two list files. The list files are used to point to all of the pharmaceuticals and all the cases that need to be loaded when the program is started. The case files store the name of the case, the items (pharmaceuticals) in the case, the category that the case belongs to, and a description of the case. The item (pharmaceutical) files contain the name to be displayed in the program, the byte value to send to the Arduino, the categories that the item belongs to, and a description of the item. In the item and case files the list of strings such as the list of categories or the description are preceded by an integer that represents the number of lines that are in that list. In the list files this is not needed because the whole file is a list and each line of the file is used.

In Future 46 we expect the program to include more functionality than we were able to implement and test. One feature that the program should include is a way to keep track of users (EMT's) and log which pharmaceuticals the user are requesting from the vending machine. To accomplish this, the program should either have its own login or tie into the system that the program resides in and use that to provide security. Another feature that the program should contain is a way of keeping track of the contents of the vending machine. Keeping track of the items in the vending machine will require additional hardware for detecting the items in the vending machine. These features aim to improve the security and reduce the amount of work needed to keep the ambulance well stocked.

3.3 Testing Methods and Metrics of Success

For any new idea to become accepted by society, it must first undergo intense scrutiny so that it is not susceptible to failure or to doubt. To ensure that the vending machine is viable in the field, several testing methods must be evaluated. Such methods should help to locate performance index that will guarantee that the product meets the metrics of success. The metrics of success for the vending machine are that it has a faster retrieval time for items and medication, causes less strain to EMS personnel, is safer than existing storage, and that the machine exposes equipment and medication to less contaminant than current methods of storage. Testing methods will consist of several comparative studies which will prove that these conditions are met. A questionnaire can be given to EMS personnel included in the study to

gather feedback about what they believe are the benefits of the vending machine as well as which aspects require more attention.

3.3.1 Retrieval Time and Stress Study

To demonstrate how the vending machine would decrease the time taken to retrieve items, a group of about 5-10 trained and experienced EMS personnel would be required. These people would comprise the experimental group which will use the vending machine in the following experiments. Because ample data exists regarding the ability of EMS to retrieve items from storage, a control group using current storage systems is not necessary and reliable data from past experiments will instead be used. The experimental group would be given between one and two hours of training to familiarize EMTs with the interface and have a good general idea of where medications and items are located on the touchscreen menu. Each member will then be asked to demonstrate their competence by participating in the actual test. In a control test, the group would be asked to retrieve items from a stationary ambulance. A second test would have the EMTs do the same, except that the ambulance would be in motion and travelling at standard city, town, rural and highway speeds. This test would show whether or not the vending machine would allow EMTs to obtain items more quickly than current storage. While this extra test could not be run until the vending machine becomes a proven method of storage in ambulances, the next logical step of testing would consist of comparing retrieval times between vending machines and cabinetry storage in an actual emergency situation. The same group would be used in the tests to confirm that the vending machine leads to less stress for EMTs personnel. Over the course of the above experiment in which the group is asked to retrieve items, data will be collected on the angles of displacement of the EMTs while retrieving items. This will be recorded on film so that joint torque can later be calculated. For example bending over 90 degrees would be an extremely strenuous position compared to standing or sitting straight up which has minimal strain associated with it. The amount of time spent in each position will be recorded for each 15 degrees of bending, starting from straight up and down all the way to 90 degrees of bending.

Materials:

Ambulance with installed vending machine.

Procedure:

a) Position EMT's in ambulance (moving or stationary depending on test)

b) Record the angles of bend (per 15 degrees) in EMT's and the time spent in those positions during item retrieval

c) Record retrieval times

Expected Results:

This study was conducted in order to determine how fast items could be retrieved as well as whether any (and if so how much) relief was being provided by using the vending machine design versus traditional storage methods. For the time taken to retrieve items when the ambulance is stationary, the expected time for withdrawing individual items using the vending machine would increase slightly (2-3 seconds) for higher numbers of items. However for cases when ambulance is in motion, significant reductions in time taken (up to nine seconds) to gather the necessary drugs would be seen (Figure 47). Although the chart does not show data for preset cases in which entire groupings of drugs could be ordered at once it is clear based on the trends shown that we would expect to see even greater decreases in time. These reductions in time would even be witnessed when the ambulance is stationary and would be expected to provide approximately 5 seconds of reduction for retrieval of five items. When the ambulance is in motion, this time reduction would be expected to jump significantly to 20 seconds reduction in time or more when using the vending machine.

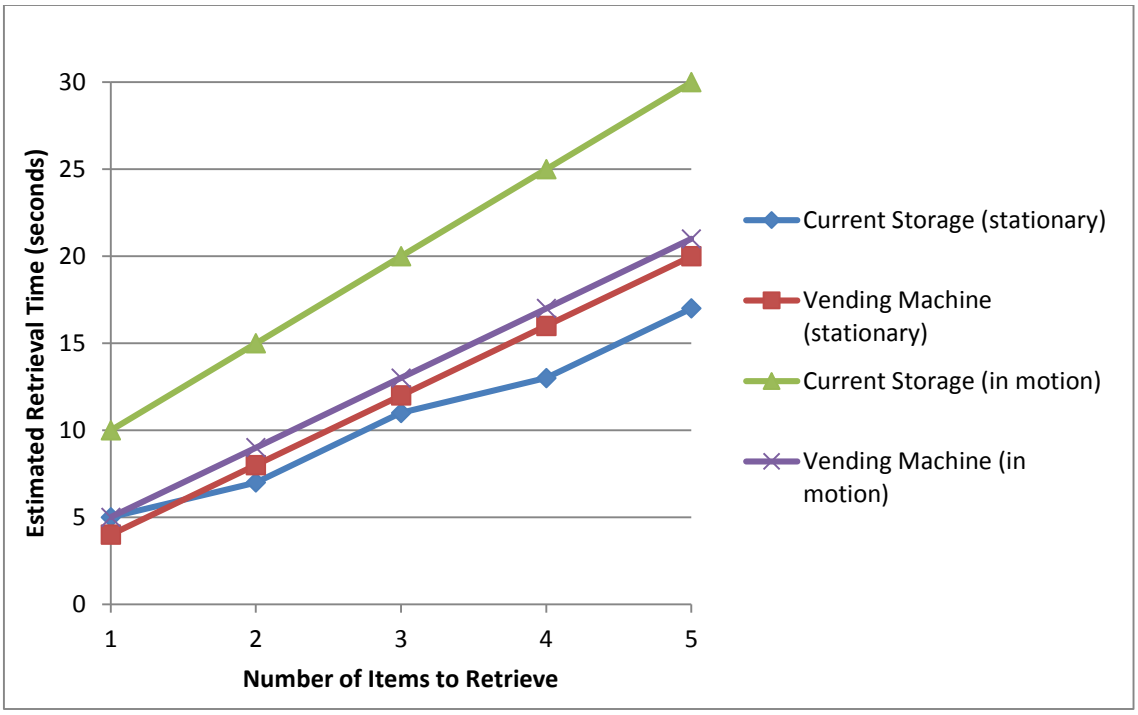

Figure 47: Estimated Retrieval Time for Varying Amounts of Items for the Two Methods of Storage

However this difference would be pretty minimal throughout the test and would only cause a loss of approximately 3 seconds for gathering 4 or 5 items. Where the vending machine truly excels is when the ambulance is in motion. The difference in retrieval time between vending machine and current storage starts at 5 seconds for a single item and increases to 9 seconds as up to five items are retrieved.

For the stress test we focus on the positions assumed during the time it takes to retrieve items from storage and not the positions assumed for treatment because the positions taken to administer treatment to patients are not expected to change. We expect to find significantly decreased instances of EMTs assuming strenuous positions in terms of degree of bending as well as shorter lengths of time spent in those positions. Figure 51 shows the expected bending of EMTs backs as well as the expected percentage time the bending is continued for. Note that this chart only displays positions during an emergency call with a patient inside the ambulance compartment. During emergency calls, most of the time spent by EMTs with a patient in the vehicle would be spent in a relatively unstrained position (<30 degrees of bending) with virtually no time spent in positions of high strain (>45 degrees of bend). This is compared to

current storage methods which force EMT's to assume high strain positions 20% of the time and in some cases EMT's may take on positions in which they will feel maximum back strain (90 degrees of bend).

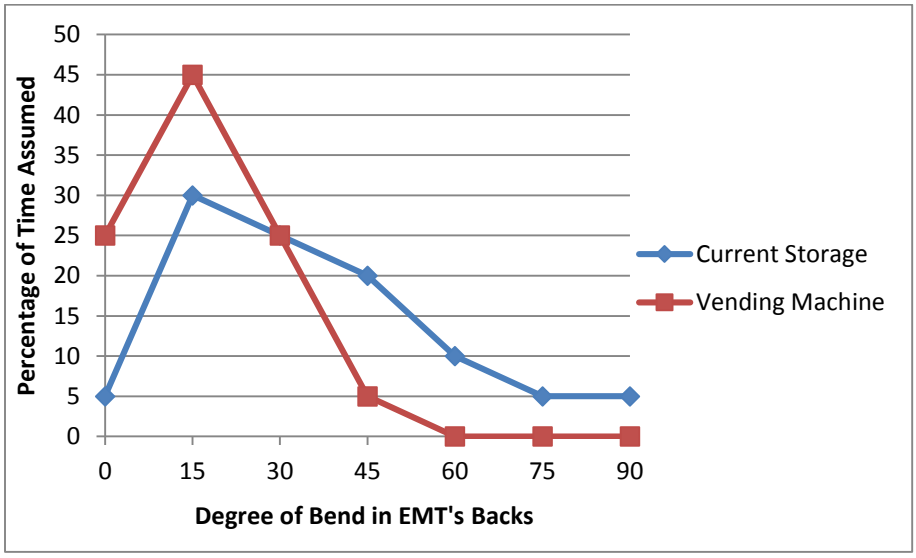

Figure 48: Time Spent by EMT's in Strained Positions

This large discrepancy between strain in current storage methods and in vending machines can be attributed to a variety of factors. The primary reason that greater bending is seen in current storage is simply because EMTs must move from the initial location at the captain's chair to obtain items from the cabinets. From the standing position, the EMT must navigate around the patient and lean towards the storage compartments to access the items all while maintaining balance (15-30 degrees). If the necessary items are located on the left side (looking from the outside in) of the ambulance the EMT will need to bend over to reach over the patient to grab the required items (45-90 degrees). All of these actions cannot be done easily or quickly (the most important aspects of providing emergency care) without getting into strained positions. Also, as explained earlier if multiple items are required then each item will have to be put down before another one can be picked up which causes additional strain from bending down to drop items and then standing back up to get new ones (30-60 degrees). When you consider the vending machine however, all that needs to be done is a small bend (15 degrees or less) to access the touchpad followed by a slightly greater bend in order to retrieve the items (30 degrees or less). Furthermore in the vending machine none of these movements

are compromised when the ambulance is in motion whereas almost all of the actions taken to retrieve items in the current storage system place the EMT's into potentially dangerous positions.

3.3.2 Loose Items Study

As a test to confirm that items are not at risk of falling, the vending machine could be accelerated and decelerated at increasing magnitudes. This would continue at increasingly higher degrees until an item fell from storage. To accomplish this, the ambulance with an installed vending machine would be driven on a closed track in the manner described above.

Materials:

Ambulance with installed fully-stocked vending machine, closed course, accelerometer.

Procedure:

a) Drive ambulance equipped with vending machine on a closed course

b) Decelerate at increasing magnitudes of acceleration until items dislodged

c) Record the acceleration when items are dislodged*

*If the items do not fall under reasonable values of deceleration then it is safe to assume that they will stay barring the case of a crash.

Expected Results:

It is expected that the items will stay in their holders at all magnitudes of acceleration. At higher magnitudes however, while the items will for the most part stay in their holders, there may be rare instances where the items are able to shift backwards or forwards one rung in the corkscrew holder which would cause some rungs to be doubly filled. Although these instances may occur, it should be noted that these events would be extremely infrequent if at all and could only be possible at very high levels of deceleration such as those experienced when the ambulance has to stop short or when it is involved in a collision. These results may show that the storage design is not optimal and could use an improved holding device.

3.3.3 Contamination Study

The final metric of success is whether items and medications stored in the vending machine would expose them to more contaminants. To prove that the vending machine does not expose pharmaceuticals and consumables to contaminants, a test must be run in which liquids are splashed onto the vending machine. If anything seeped into the machine the location would be recorded and improvements to the vending machine would be made.

Materials:

Vending machine, various liquids.

Procedure:

a) Throw liquids on to vending machine

b) Detect for penetration by liquids

c) If there is, record where penetration occurred and what volume entered the machine

Expected Results:

This study was conducted in order to determine whether or not the vending machine was permeable to liquid contaminants It is expected that the items inside will be completely free from any form of liquid contamination including bacteria as well as whichever liquids are used in the testing procedure. Because of the strict guidelines set forth by the many ambulance governing agencies, for the vending machine design to even be remotely viable, it would have to be able to effectively keep contaminants out. Figure 49 shows the expected relative amounts of contamination for each system of storage after the liquid test was performed. It also shows how much relative contamination is expected to be seen over the course of normal use within the ambulance for each system.

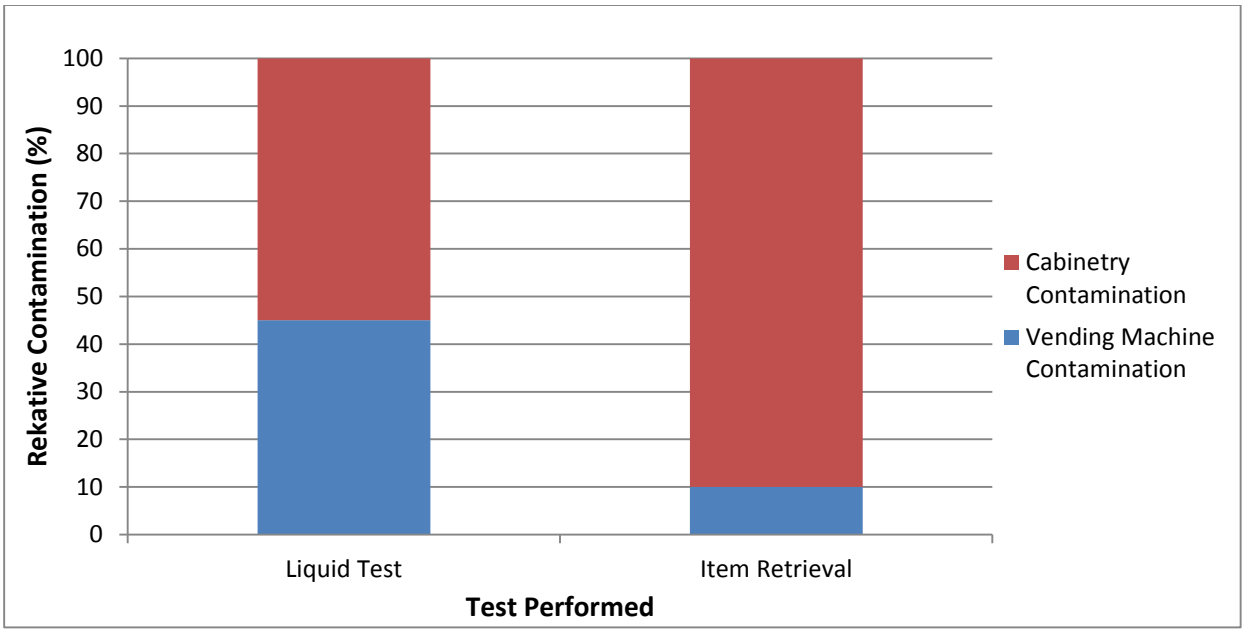

Figure 49: Relative Contamination between the Two Storage Methods

As shown in the figure, both systems would fair equally well in keeping out contaminants when liquids were splashed onto them. However, over the course of normal use of the vending machine, it would be far better at keeping out contamination. The cabinetry storage system would have upwards of nine times as much contamination as the vending machine.

3.3.4 Survey

In addition to the proposed tests, the members of the experimental group would be asked to answer a questionnaire regarding use of the vending machine and how it compares to their experience using current storage methods. The purpose of these questions is to gain a better understanding of how EMS feel about embracing this new technology as well as seeking to improve the quality of the vending machine based on input from the people who would be actually using it. Questions may include but would not be limited to the following:

1) Which method do you prefer?

2) Which method is easier to use?

3) Which method is less strenuous to use?

4) Which method would you trust more while you are on the job?

5) Is there anything about the machine that is difficult to use? If so what?

6) Is the vending machine placed conveniently? If not, where would it be optimally located?

Further questions would be focused on gaining more insight about how EMTs prefer to operate the vending machine. Questions would also address how the vending machine could most seamlessly fit into the standard of prehospital care while at the same time providing better levels of efficiency, relief, safety and cleanliness than current methods of item storage in ambulances. Based on EMT responses to the questionnaire, changes to the vending machine could be made in order to make it more compatible with the work environment that EMTs will be operating in.

CHAPTER 4. Concluding Remarks

Emergency medical care is an extremely important part of society. Since the 1800's ambulances have been the vehicle of choice for transporting patients to a location where they can receive proper medical attention. Over time ambulances changed their approach by not only transporting patients but also providing emergency care until they reached their destination. Although this was an overwhelmingly positive change that has led to the saving of countless lives, it has also led to many problems for EMTs who work inside the ambulances. Pharmaceuticals and consumables which are necessary for the treatment of patients are stored inefficiently in cabinets. This system of storage forces EMTs to overextend themselves, and thereby leads to unnecessary strain which, over time, can cause injury. Furthermore items are not securely stored in cabinets and may fall out and become dangerous projectiles. This project sought to replace the current storage system used within ambulances with a vending machine.

The vending machine was designed with the intent to more securely hold items, be reliable, easily restocked, have a convenient and intuitive interface, dispense items to an accessible location, and be durable and easy to repair. By using a corkscrew dispenser the vending machine is able to keep items firmly in place even when the ambulance is accelerating. The dispenser is designed in such a way that only one item will drop per request so that no additional time is wasted dealing with multiple items or having to reorder an item. To do so however, solid, liquid and inhaled items must be prepackaged in blister packs, prefilled syringes and single-dose inhalers respectively. Interfacing with EMTs is done through a touch screen and is intuitive, meaning that EMTs will be able to quickly learn how to use it. The conveyor belt within the vending machine makes it easy for the items dropped by the dispensers to get to a location close to the captain's chair. Access panels to the main storage area as well as panels to the wiring and battery mean that the machine can be easily repaired if anything goes wrong. All of these beneficial attributes make the vending machine concept a revolutionary idea that could change the way emergency medical care is run.

Although the scope of this project was not broad enough nor was there enough time to consider all possibilities for the vending machine, several areas of improvement have been identified for future groups to expand upon. As of now the vending machine may be accessed by anyone because a security system was not installed. A security system for the interface would ideally require at minimum a password to access as well as a way to document specifically which EMT was "logged in". This way if items went missing someone could be held accountable. Currently it is difficult to determine how many of the items are still in the dispensers without removing the dispensers. An upgrade to the vending machine would be to have it keep track of the number of pharmaceuticals or consumables left in the dispensers. This number could either be displayed on the interface or a light could turn on if there was less than a certain percentage remaining. The vending machine cannot be so heavy that it overburdens the ambulance nor makes the ambulance becomes side-heavy. Because of these stipulations, the materials used to create the vending machine must be chosen prudently. Such considerations as these must be taken seriously in order to make the vending machine a viable option for use in ambulances.

EMTs are disproportionately incurring work-related injuries compared to other emergency care providers. These injuries, mostly to the back and shoulders, are a direct result of the inefficient storage utilized in modern ambulances. Dramatic changes in the storage system are necessary in order to ensure the safety of the emergency medical care professionals we rely upon. Incorporation of an automated dispensing machine into the patient compartment of the ambulance will greatly ease the stress on EMTs. Paramedics will no longer be forced to assume strenuous positions in order to retrieve items. Additionally, items will be unable to fall out from their location into a storage container and therefore cannot become dangerous to paramedics and patients alike. As a consequence of the vending machine, the injury rates of paramedics and EMTs will be expected to decrease significantly and will hopefully be more in line with those of other emergency professions.

References

[1] "History of Ambulances." History of Ambulances. N.p., n.d. Web. 11 May 2014. <http://www.emt-resources.com/History-of-Ambulances.html>.

[2] "The Ford Model T Specifications." *The Ford Model T Specifications*. N.p., n.d. Web. 11 May 2014. <http://www.barefootsworld.net/ford-t-specs.html>.

[3] "Factsheets : Ford Model T Ambulance." *Factsheets : Ford Model T Ambulance*. N.p., n.d. Web. 11 May 2014. <http://www.nationalmuseum.af.mil/factsheets/factsheet.asp?id=983>.

[4] Morette, Kyle Andrew et al. *History and Development of Emergency Transportation*. Worcester, MA: Worcester Polytechnic Institute, 2011. Print.

[5] "KKK-A-1822F." *KKK-A-1822F*. N.p., n.d. Web. 11 May 2014. <http://www.deltaveh.com/KKK-A-1822F.htm>.

[6] "Braun Ambulances - Custom Ambulances Built for Life." *Braun Ambulances - Custom Ambulances Built for Life*. N.p., n.d. Web. 11 May 2014. <http://www.braunambulances.com/>.

[7] "Welcome to American Emergency Vehicles." *Welcome to American Emergency Vehicles*. N.p., n.d. Web. 11 May 2014. <http://www.aev.com/>.

[8] “Number of nonfatal occupational injuries and illnesses involving days away from work by occupation and selected parts of body affected by injury or illness, 2004,” No. R10, *Occupational injury and illness*. Bureau of Labor Statistics, 2004.

[9] The U.S. Bureau of labor Statistics. (2010). Occupational Outlook Handbook. Retrieved from <http://www.bls.gov/ooh/healthcare/registered-nurses.htm>

[10] The U.S. Bureau of labor Statistics. (2010). Occupational Outlook Handbook. Retrieved from <http://www.bls.gov/ooh/healthcare/emts-and-paramedics.htm>

[11] “Number of nonfatal occupational injuries and illnesses involving days away from work by occupation and selected parts of body affected by injury or illness, 2011,” No. R10, *Occupational injury and illness*. Bureau of Labor Statistics, 2011.

[12] “Number of nonfatal occupational injuries and illnesses involving days away from work by occupation and selected sources of injury or illness, private industry, 2011,” No. R10, *Occupational injury and illness*. Bureau of Labor Statistics, 2011.

[13] Studnek, Jonathan R., J. Mac Crawford, and Antonio R. Fernandez. "Evaluation of occupational injuries in an urban emergency medical services system before and after implementation of electrically powered stretchers." *Applied Ergonomics* 43.1 (2012): 198-202.

[14] Maguire, Brian J., et al. "Occupational injuries among emergency medical services personnel." *Prehospital Emergency Care* 9.4 (2005): 405-411.

[15] Becker, Les R., et al. "Relative risk of injury and death in ambulances and other emergency vehicles." *Accident Analysis & Prevention* 35.6 (2003): 941-948.

[16] Ferreira, Jeremy, and Sue Hignett. "Reviewing ambulance design for clinical efficiency and paramedic safety." *Applied Ergonomics* 36.1 (2005): 97-105.

[17] Boocock, Mark G., Mike I. Gray, and Sally Williams. "Patient handling in the ambulance services: Case study investigations." *Contemporary Ergonomics* (2002): 33-38.

[18] Letendre, J., Robinson, D., 2000. Evaluation of paramedic's tasks and equipment to control the risk of musculoskeletal injury. Ambulance Paramedics of British Columbia, CUPE Local 873 Internal Report 6-08-0793.

[19] Rodgers, L. M. "A five year study comparing early retirements on medical grounds in ambulance personnel with those in other groups of health service staff Part II: Causes of retirements." *Occupational Medicine* 48.2 (1998): 119-132.

[20] Doormaal, M. T. A. J., et al. "Physical workload of ambulance assistants." *Ergonomics* 38.2 (1995): 361-376.

[21] Hirshon, Jon M., MD, MPH, PhD, FACEP. "ACEP 2014 EM Report Card." *ACEP 2014 EM Report Card*. American College of Emergency Physicians, 2014. Web. 08 Apr. 2014.

[22] Perron, Jeffrey M. et al. "SLATE Ambulance: Designing an Interior Storage System for Improved Sanitation." Mirad Laboratory and Worcester Polytechnic Institute, May 31, 2013.

[23] General Services Administration, "Federal Specification for the Star-of-Life Ambulance," *KKK-A-1822f*, August 28, 2012.

[24] "Paramedic emergency ambulance staff, medications, equipment, and supplies," County of San Mateo, San Mateo, CA, April 1, 2002.

[25] B. Crichton, "Keep in a cool place: exposure of medicines to high temperatures in general practice during a British heatwave," Jrsm, vol. 97, pp. 328-329, 2004.

[26] (2007). PharmGuard, An Environmental Control System for EMS Pharmaceutical Storage. Available: http://www.pharmguard.com/pharmguard.htm

[27] United States of America. Food and Drug Administration. *Controlled Substances Act*. U.S. Food and Drug Administration, 11 June 2009. Web. 6 Jan. 2014.

[28] Taggart, Starr and Cecie Starr. *Biology: The Unity and Diversity of Life*. California: Wadsworth, 1939: 398

[29] "Frequently Asked Questions." *Japan Vending Machine Manufacturers Association*. Japan Vending Machine Manufacturers Association, 1999. Web. 6 Jan. 2014.

[30]"Automatic Vending Machines." *Business.highbeam.com*. HighBeam Business, n.d. Web. 06 Jan. 2014.

[31] Jaffe, Eric. "Smithsonian.com." *Smithsonian Magazine*. Smithsonian, Dec. 2006. Web. 06 Jan. 2014.

[32] "Vending History." *Variety FoodServices*. Variety FoodServices, n.d. Web. 6 Jan. 2014.

[33] "The Vending Revolution." *Aimsuite.intel.com*. Intel, 27 July 2012. Web. 06 Jan. 2014. <https://aimsuite.intel.com/vending-revolution>.

[34] Mann, Dave. "2001 Darwin Award: Coke Is It!" *Darwinawards.com*. Darwin Awards, n.d. Web. 06 Jan. 2014. <http://darwinawards.com/darwin/darwin2001-25.html>.

[35] "Title 68: Professions and Occupations Chapter VII: Department of Financial and Professional Regulation Subchapter B: Professions and Occupations Part 1330 Pharmacy Practice Act Section 1330.730 Drug Prepackaging." *Illinois General Assembly*. N.p., n.d. Web. 08 Apr. 2014. <http://www.ilga.gov/commission/jcar/admincode/068/068013300G07300R.html>.

Appendix A: Report Card Grading Criteria for Each of the Five Categories

INDICATOR WEIGHTS BY METRIC

Indicator weights are presented as a percentage of the total category.

ACCESS TO EMERGENCY CARE 30% OF THE OVERALL STATE GRADE	
Access to providers 25.00% of the category	
Board-certified emergency physicians per 100,000 pop.	3.57
Emergency physicians per 100,000 pop.	3.57
Neurosurgeons per 100,000 pop.	0.89
Orthopedists and hand surgeon specialists per 100,000 pop.	0.89
Plastic surgeons per 100,000 pop.	0.89
ENT specialists per 100,000 pop.	0.89
Registered nurses per 100,000 pop.	7.14
Additional primary care FTEs needed	3.57
Additional mental health FTEs needed	3.57
Access to treatment centers 25.00% of the category	
Level I or II trauma centers per 1M pop.	5.00
% of population within 60 minutes of Level I or II trauma center	5.00
Accredited chest pain centers per 1M pop.	5.00
% of population with an unmet need for substance abuse treatment	5.00
Pediatric specialty centers per 1M pop.	5.00
Financial barriers 25.00% of the category	
Physicians accepting Medicare per 100 beneficiaries	5.00
Medicaid fee levels for office visits as a % of the national average	2.50
% change in Medicaid fees for office visits (2004-05 to 2007)	2.50
% of adults with no health insurance	5.00
% of children with no health insurance	5.00
% of adults with Medicaid	5.00
Hospital capacity 25.00% of the category	
Emergency departments per 1M pop.	3.00
Hospital closures in 2006	3.00
Staffed inpatient beds per 100,000 pop.	10.00
Hospital occupancy rate per 100 staffed beds	3.00
Psychiatric care beds per 100,000 pop.	3.00
State collects data on diversion	3.00
Total*	**99.98**

QUALITY AND PATIENT SAFETY ENVIRONMENT 20% OF THE OVERALL STATE GRADE	
State Systems 66.67% of the category	
Funding for quality improvement within the EMS system	6.35
Funded state EMS medical director	6.35
Emergency medicine residents per 1 million pop.	3.18
Adverse event reporting requirement	6.35
Hospital-based infection reporting requirement	6.35
Mandatory quality reporting requirement	6.35
% of counties with Enhanced 911 capability	6.35
Uniform system for providing pre-arrival instructions	6.35
State has or is working on a stroke system of care	6.35
State has or is working on a PCI network or a STEMI system of care	6.35
Statewide trauma registry	6.35
Institution 33.33% of the category	
% of hospitals with computerized practitioner order entry	6.67
% of hospitals with electronic medical records	6.67
% of patients with acute myocardial infarction given PCI within 90 minutes of arrival	13.33
Number of JCAHO reviewed sentinel events per 1 million pop. (1995-2006)	6.67
Total*	**100.02**

INDICATOR WEIGHTS BY METRIC

Indicator weights are presented as a percentage of the total category.

MEDICAL LIABILITY ENVIRONMENT 20% OF THE OVERALL STATE GRADE	
Legal atmosphere 25.00% of the category	
Lawyers per 10,000 pop.	0.00
Lawyers per physician	0.00
Lawyers per emergency physician	0.00
ATRA judicial hellholes (range 0 to -7)	5.00
Malpractice award payments per 100,000 pop.	4.00
Average malpractice award payments	4.00
Databank reports per 1,000 physicians	4.00
Patient compensation fund	4.00
Health court pilot project grant	4.00
Insurance availability 20.00% of the category	
Number of insurers writing medical liability policies per 1,000 physicians	5.00
Average medical liability insurance premium for primary care physicians	7.50
Average medical liability insurance premiums for specialists	7.50
Tort reform 55.00% of the category	
Presence of pretrial screening panels	3.33
Are findings admissible as evidence?	1.67
Periodic payments	5.00
Medical liability cap on non-economic damages	20.00
Additional liability protection for EMTALA-mandated emergency care	5.00
Joint and several liability abolished	5.00
State provides for case certification	5.00
Expert witness required to be of the same specialty as the defendant	5.00
Expert witness must be licensed to practice medicine in the state	5.00
Total*	**100.00**

PUBLIC HEALTH AND INJURY PREVENTION 15% OF THE OVERALL STATE GRADE	
Traffic safety and drunk driving 27.80% of the category	
Traffic fatalities per 100,000 pop.	5.56
% of traffic fatalities alcohol related	5.56
Front occupant restraint use (%)	5.56
Helmet use required for all motorcycle riders	2.78
Child safety seat/seat belt legislation (10 points possible)	2.78
Immunization 16.70% of the category	
% of children immunized, aged 19–35 months	5.56
% of adults aged 65+ who received flu vaccine in the last 12 months	5.56
% of adults aged 65+ who ever received pneumococcal vaccine	5.56
Injury 16.70% of the category	
Fatal occupational injuries per 1M workers	2.78
Homicides and suicides (non-motor vehicle) per 100,000 pop.	5.56
Unintentional fall-related fatal injuries per 100,000 pop.	2.78
Unintentional fire/burn-related fatal injuries per 100,000 pop.	2.78
Unintentional firearm-related fatal injuries per 100,000 pop.	2.78
State injury prevention efforts 22.20% of the category	
Gun-purchasing legislation (8 points possible)	5.56
% of tobacco settlement funds spent on health-related services and programs	5.56
Total injury prevention funds per 1,000 pop.	2.78
Unintentional injury prevention funds per 1,000 pop.	2.78
Intentional injury prevention funds per 1,000 pop.	2.78
Fall injury prevention funds per 1,000 pop.	2.78
Health risk factors 16.70% of the category	
Infant mortality rate per 1,000 live births	5.56
% of adults with BMI > 30	5.56
Current smokers, % of adults	5.56
Binge alcohol drinkers, % of adults	5.56
Total*	**100.08**

Appendix B: Report card for Massachusetts

Access To Emergency Care	B
Board-certified emergency physicians per 100,000 pop.	14.2
Emergency physicians per 100,000 pop.	19.7
Neurosurgeons per 100,000 pop.	2.8
Orthopedists and hand suregeon specialists per 100,000 pop.	12.7
Plastic surgeons per 100,000 pop.	3.3
ENT specialists per 100,000 pop.	4.2
Registered nurses per 100,000 pop.	1317.4
Additional primary care FTEs needed per 100,000 pop.	0.7
Additional mental health FTEs needed per 100,000 pop.	0.3
% of children able to see provider	97.3
level I or II trauma centers per 1M pop.	1.4
% of population within 60 minutes of Level I or II trauma center	93
Accredited chest pain centers per 1M pop.	0.2
% of population with an unmet need for substance abuse treatment	9.9
Pediatric specialty centers per 1M pop.	2.1
Physicians accepting medicare per 100 beneficiaries	4.1
Medicaid fee levels for office visits as a % of the national avergae	107.1
% change in Medicaid fees for office visits (2007 to 2012)	17.8
% of adults with no health insurance	3.8
% of adults underinsured	6.9
% of children with no health insurance	2.5
% of children underinsured	17.5
% of adults with Medicaid	17.7
Emergency departments per 1M pop.	9.8
Hospital closures in 2011	1
Staffed inpatient beds per 100,000 pop.	321.2
Hospital occupancy rate per 100 staffed beds	75
Psychiatric care beds per 100,000 pop.	27.4
Median minutes from ED arrival to ED departure for admitted patients	311
State collects data on diversion	N/A
Medical Liability Environment	D-
Lawyers per 10,000 pop.	24.5
Lawyers per physician	0.5

Lawyers per emergency physician	12.4
ATRA judicial hellholes (range 2 to -6)	0
Malpractice award payments per 100,000 pop.	1.4
Average malpractice award payments	$519,961
Databank reports per 1,000 physicians	17.3
Provider apology is inadmissible as evidence	Yes
Patient compensation fund	No
Number of insurers writing medical liability policies per 1,000 physicians	2.4
Average medical liability insurance premium for primary care physicians	$15,235
Average Medical liability insurance premium for specialists	$77,668
Prescence of pretrial screening panels	Mandatory
Pretrial screening panel's findings admissible as evidence	Yes
Periodic payments	No
Medical liability cap on non-economic damages	>$500,000
Additional liability protection for EMTALA-mandated emergency care	No
Joint and several liability abolished	No
Collateral source rule, provides for awards to be offset	yes
State provides for case certification	No
Expert witness must be of the same specialty as the defendant	No
Expert witness must be licensed to practice medicine in the state	No
Quality & Patient Safety Environment	B+
Funding for quality improvement within EMS system	No
Funded state EMS medical director	Yes
Emergency medicine residents per 1M pop.	33.1
Adverse event reporting required	Yes
% of counties with E*911 capability	100
Uniform system for providing pre-arrival instructions	No
CDC guidelines are basis for state field triage protocols	Yes (2011)
State has or is working on a stroke system of care	Yes
Triage and destination policy in place for stroke patients	Yes
State has or is working on a PCI network or a STEMI system of care	Yes
Triage and destination policy in place for STEMI patients	Yes
Statewide trauma registry	Yes
Triage and destination policy in place for trauma patients	Yes
Prescription drug monitoring program (range 0-4)	3

% of hospitals with computerized practitioner order entry	92.7
% of hospitals with electronic medical records	97.5
% of patients with AMI given PCI within 90 minutes of arrival	95
Median time to transfer to another facility for acute coronary intervention	99
% of patients with AMI who received aspirin within 24 hours	100
% of hospitals collecting data on race/ethnicity and primary language	52.3
% of hospitals having or planning to develop a diversity strategy/plan	46.5
Public Health & Injury Prevention	A
Traffic fatalities per 100,000 pop.	3.8
Bicyclist fatalities per 100,000 cyclists	1.9
Pedestrian fatalities per 100,000 pedestrians	2.1
% of traffic fatalities alcohol related	39
Front occupant restraint use (%)	73.2
Helmet use required for all motorcycle riders	Yes
Child safety seat/seat belt legislation (range 0-10)	5
Distracted driving legislation (range 0-4)	2
Graduated drivers' license legislation (range0-5)	1
% of children immunized, aged 19-35 months	80.3
% of adults aged 65+ who received flu vaccine in past year	66.9
% of adults aged 65+ who ever received pneumococcal vaccine	72.2
Fatal occupational injuries per 1M workers	16.5
Homocides and suicides (non-motor vehicles) per 100,000 pop.	11.2
Unintentional fall-related fatal injuries per 100,000 pop.	7.9
Uninterntional fire/burn-related fatal injuries per 100,000 pop.	0.5
Unintentional firearm-related fatal injuries per 100,000 pop.	< 0.1
Unintentional poisoning-related fatal injuries per 100,000 pop.	10.5
Total injury prevention funds per 1,000 pop.	$2,950.94
Dedicated child injury prevention funding	Yes
Dedicated elderly injury prevention funding	Yes
Dedicated occupational injury prevention funding	No
Gun-purchasing legislation (range 0-6)	4
Anti-smoking legislation (range0-3)	3
Infant mortality rate per 1,000 live births	4.4
Binge alcohol drinkers, % of adults	20.6
Current smokers, % of adults	18.2

% of adults with BMI > 30	22.7
% of children obese	14.5
Cardiovascular disease disparity ratio	2.5
HIV diagnoses disparity ratio	NR
Infant mortality disparity ratio	2.7
Disaster Preparedness	C
Per capita federal disaster preparedness funds	$8.54
State budget line item for health care surge	NR
ESF-8 plan shared with all EMS and essential hospital personnel	Yes
Emergency physician input into the state planning process	Yes
Public health and emergency physician input during an ESF-8 response	Yes
Drills, excersises conducted with hospital personnel, equipment, facilities per hospital	0.2
Accredited by the Emergency Management Accreditation Program	Conditionally
Special needs patients in medical response plan	Yes
Patients on medication for chronic conditions in medical response plan	Yes
Medical response plan for supplying dialysis	Yes
Mental health patients in medical response plan	Yes
Medical response plan for supplying psychotropic medication	Yes
Mutual aid agreements with behavioral health providers	State-level
Long-term care and nursing home facilities must have written disaster plan	Yes
State able to report number of exersises with long-term care or nursing home facilities	Yes
"Just-in-time" training systems in place	Statewide
Statewide medical communication system with one layer of redundancy	Yes
Statewide patient tracking system	No
Statewide real-time or near real-time syndromic surveillance system	Yes
Real-time surveillance system in place for common ED presentations	In metro areas
Bed surge capacity per 1M pop.	245.6
ICU beds per 1M pop.	245.4
Burn unit beds per 1M pop.	10.5
Verified burn centers per 1M pop.	0.5
Physicians in ESAR*VHP per 1M pop.	87.6
Nurses in ESAR*VHP per 1M pop.	537.2
Behavioral health professionals in ESAR*VHP per 1M pop.	50.9
Strike teams or medical assistance teams	Yes
Disaster training required for essential hospital, EMS personnel	No, Yes

Liability protections for health care workers during a disaster (range 0-4)	1
% of RNs received disaster training	35.4

Appendix C: Complete listing of equipment carried in ambulances[24]

Name	Amount
Box of Gloves	6
.9% NaCl, Baxter	2
.9% NaCl, Baxter, Large	4
Bladder Irrigation Set	2
Sterile Burn Sheet	5
Continu-Flo Solution Set	1
IV Bags	8
BP Cuff	1
KED Bag, Green	1
Backboard, Blue	1
Sharps Container, Small	3
Bag of Splints	1
Femur Traction Splint	3
Transfer Sheet	1
Pillow	1
Fire Extinguisher	1
Bed Pan	1
Cardboard Bin	51
Medline Tissue Box	6
Eye Pads	17
Adult BP Cuff	1
Child and Infant Cuffs	2
Cravats	20
Padded Boards	10
OB Pads	7
Albuterol	1
Insta-Glucose	4
Tubing	5
Epi-Pen	3
Naloxone Hydrochloride	8
MAD Nasal	12
Aspirin	2
Albuterol Tube	41
Coban Wrap	5
Large Ace Wrap	
Small Ace Wrap	
5X9 Gauze, Small	12
5X9 Gauze, Large	19
Kerlix	6
Kendall Conform Bandage	10
4X4 Sponges	44
Petroleum Gauze	5
Over Wraps	7
Mattress Porta Warm	2
Multi-trauma Dressing	6

Disposable Obstetrics Kit	3
Trauma Numbers	20
Poison Kits	2
Stretcher Straps	3
Dispatch Wio=pes	1
Alcohol Bottles	2
Tubing and Scissors	3
Hydrogen Peroxide Bottles	2
Heat Packs	24
Cold Packs, Kimberly Clark	11
Vomit/Urine Bag	17
Swivel Adapter	5
Scissors	16
Pupil Gauges	10
Trauma Shears	2
B and F	4
Baby Shampoo	2
Cloth Tape	2
Cloth Tape, Large	2
Transparent Tape	4
Transparent Tape, Large	4
Seatbelt Guard	3
Trash Bag	1
Suction Bags	2
Red Hazard Bags	2
Purell Refill	1
AirLife Bubble Humidifier	1
Blue Trash Bag	1
Dispatch Wipes	1
Suction Tube	2
Suction Handle	3
Catheter	7; 1,2,2,2
Oral Airways	37; 3,5,5,5,6,4,9
Adult Nasal Airways	10
Pedi Nasal Airways	1
Alcohol Wipes	1
Trach. Mask	5
Bandaids	full bin
Finger Pricks	225
Probe Covers	42
Glucose Kit	1
Test Strips	3
Pedi Non-Rebreathing Mask	1
Allergic Reaction	3
Tubing, Breathing	2
Adult Breathing Kit	5
Adult Aerosol Mask	3
O2 Tubing	7

Adult NRB	5
Adult Nasal Cannula	4
Pedi Nasal Cannula	8
Pedi NRB	5
Particulate Respirator	2
Procedure Mask	1
Tubing, Epinephrine	1
Rescucitator	1
Infection Control Kit	3
Speed Sheet	1
Cups	1 tube
Plastic Wrap	1
Emergency Blanket	2

Appendix D: Complete list of medications carried in San Mateo County, California ambulances

Medication	Minimum Quantity	Standard Quantity
Adenosine	30 mg	60 mg
Albuterol 0.83% solution	1 dose	4 doses
Aspirin: children's chewable	1 bottle	2 bottles
Atropine 1.0 mg/5 ml preload	4 mg	6 mg
Benadryl 50 mg/1 ml	100 mg	200 mg
Charcoal Slurry 25 Grams/120 ml	1 dose	2 doses
Calcium Chloride 1 gm/10 ml preload	1 gm	1 gm
Dextrose 50% 25 gm/50 ml preload	50 gm	100 gm
Dopamine 400 mg vial	400 mg	800 mg
Epinephrine 1:1,000 1 mg/ml ampule	2 mg	4 mg
Epinephrine 1:10,000 1 mg/10 ml preload	4 mg	8 mg
Epinephrine 1:1,000 30 ml vial	1	2

Medication	Minimum Quantity	Standard Quantity
Glucagon 1 mg/vial	1 mg	2 mg
Glucola 10 oz. or Glucose Paste, tube	2	2
Lasix 40 mg	120 mg	160 mg
Lidocaine 100 mg/5 ml preload	400 mg	800 mg
Narcan 2.0 mg/2 ml preload	2.0 mg	8.0 mg
Nitroglycerine 0.4 mg tab or Nitroglycerine metered spray	2.4 mg 2 bottles	 3 bottles
Infant Sodium Bicarbonate 5 mEq/10 ml preload	10 mEq	20 mEq
Sodium Bicarbonate 50 mEq/50 ml preload	100 mEq	200 mEq
Morphine Sulfate 10 mg/1 ml ampule w/ tubex carpuject	20 mg	50 mg
Versed 2mg / 2cc vial, ampule, or preload	4 mg	8 mg
Medication Labels	10	10
Portable Medication Box	1	1

IV Fluids	Minimum Quantity	Standard Quantity
Normal Saline 250 ml	2 each	8
Normal Saline 1000 ml	4 each	10
Normal Saline 10 ml vial	4 each	10

Appendix E: Grouping of Medications Based on Function and Method of Administration

Drug	Group	Primary Function	Method of Administration
Adenosine	Cardiovascular	Antiarrhythmic Agent	IV
Amiodarone	Cardiovascular	Antiarrhythmic Agent	Oral or IV
Magnesium Sulfate	Cardiovascular	Antiarrhythmic Agent	IV
Procainamide	Cardiovascular	Antiarrhythmic Agent	IV
Atropine	Cardiovascular	Anticholinergic Agent	IV
Nitroglycerin Tab or Spray	Cardiovascular	Vasodilator	Sublingual or IV
Dobutamine	Cardiovascular	Sympathomimetic	IV
Dopamine	Cardiovascular	Sympathomimetic	IV
Epinephrine 1:10000	Cardiovascular	Sympathomimetic	IV
Diltiazem	Cardiovascular	Treats High Blood Pressure (Calcium Channel Blocker)	Oral
Verapamil	Cardiovascular	Treats High Blood Pressure (Calcium Channel Blocker)	Oral or IV
Captopril	Cardiovascular	Treats High Blood Pressure (ACE Inhibitor)	Oral
Enalapril	Cardiovascular	Treats High Blood Pressure (ACE Inhibitor)	IV
Aspirin	Cardiovascular	Analgesic, Anti-Inflammatory	Oral
Epinephrine 1:1000	Cardiopulmonary	Sympathomimetic	IV
Albuterol	Cardiopulmonary	Bronchodilator	Nebulized Inhalation
Terbutaline	Cardiopulmonary	Bronchodilator	Inhalation
Ipratropium Bromide	Cardiopulmonary	Anticholinergic Agent	Inhalation
Diphenhydramine	Cardiopulmonary	Antihistamine	Oral
Lasix (Furosemide)	Cardiopulmonary	Treats High Blood Pressure (Diuretic)	Oral
Lorazepam	Analgesics/ Anesthetics	Anesthetic	Oral or IM
Nitrous Oxide	Analgesics/ Anesthetics	Anesthetic	Inhalation
Benzocaine	Analgesics/ Anesthetics	Local Anesthetic	Topical

Lidocaine	Analgesics/ Anesthetics	Local Anesthetic	Topical
Tetracaine	Analgesics/ Anesthetics	Local Anesthetic	Topical
Diazepam	Analgesics/ Anesthetics	Antiepileptic	IM
Midazolam	Analgesics/ Anesthetics	Antiepileptic	IM
Fentanyl	Analgesics/ Anesthetics	Analgesic	IM
Morphine Sulfate	Analgesics/ Anesthetics	Analgesic	IV
Dexamethasone Sodium Phosphate	Anti-inflammatory/ Immunosuppressives	Anti-Inflammatory	IV
Methylprednisolone	Anti-inflammatory/ Immunosuppressives	Anti-Inflammatory	IV
Hydrocortisone Sodium Succinate	Anti-inflammatory/ Immunosuppressives	Immunosuppresive Drug	IV
Glucagon	Miscellaneous	Hyperglycemic Agent	IM
Glucola or Glucose Paste	Miscellaneous	Hyperglycemic Agent	Oral
Calcium Chloride	Miscellaneous	Hypercalcemic Agent	IV
Calcium Gluconate	Miscellaneous	Hypercalcemic Agent	IV
Pralidoxime CL	Miscellaneous	Organophosphate Antidote	IV
Sodium Thiosulfate	Miscellaneous	Cyanide Antidote	IV
Narcan (Naloxone)	Miscellaneous	Opioid Antagonist	IV
Charcoal Slurry	Miscellaneous	Adsorbent	Oral
Sodium Bicarbonate	Miscellaneous	Used to Basify the Blood (counters acidosis)	IV
Sodium Bicarbonate (infant)	Miscellaneous	Used to Basify the Blood (counters acidosis)	IV
Ondansetron	Miscellaneous	Antiemetic	Oral or Rectal
Oxytocin	Miscellaneous	Facilitates Childbirth	Intranasal or IV

Appendix F: Engineering Drawings of Functional Elements

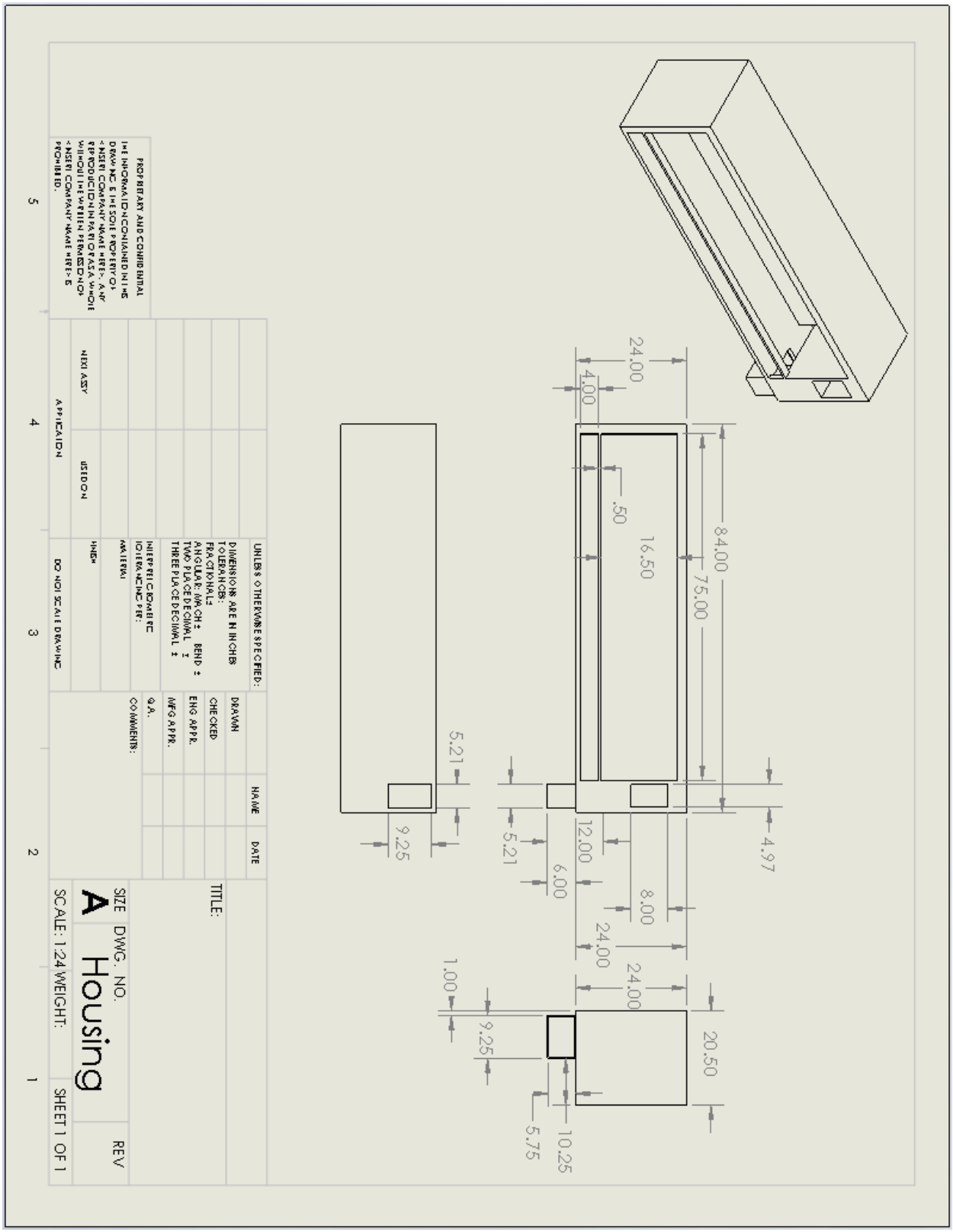

Figure 50: Housing

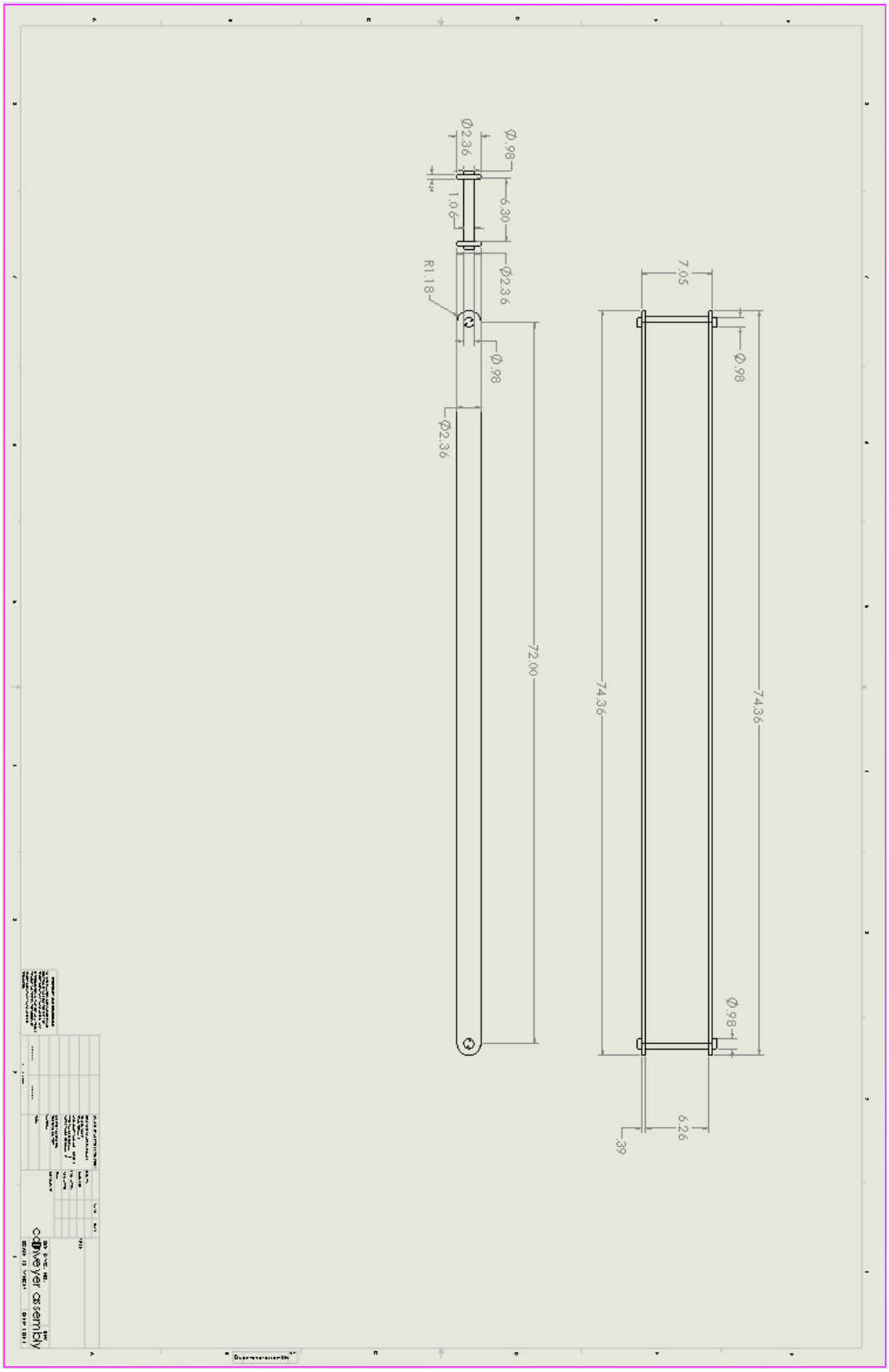

Figure 51: Conveyor Belt

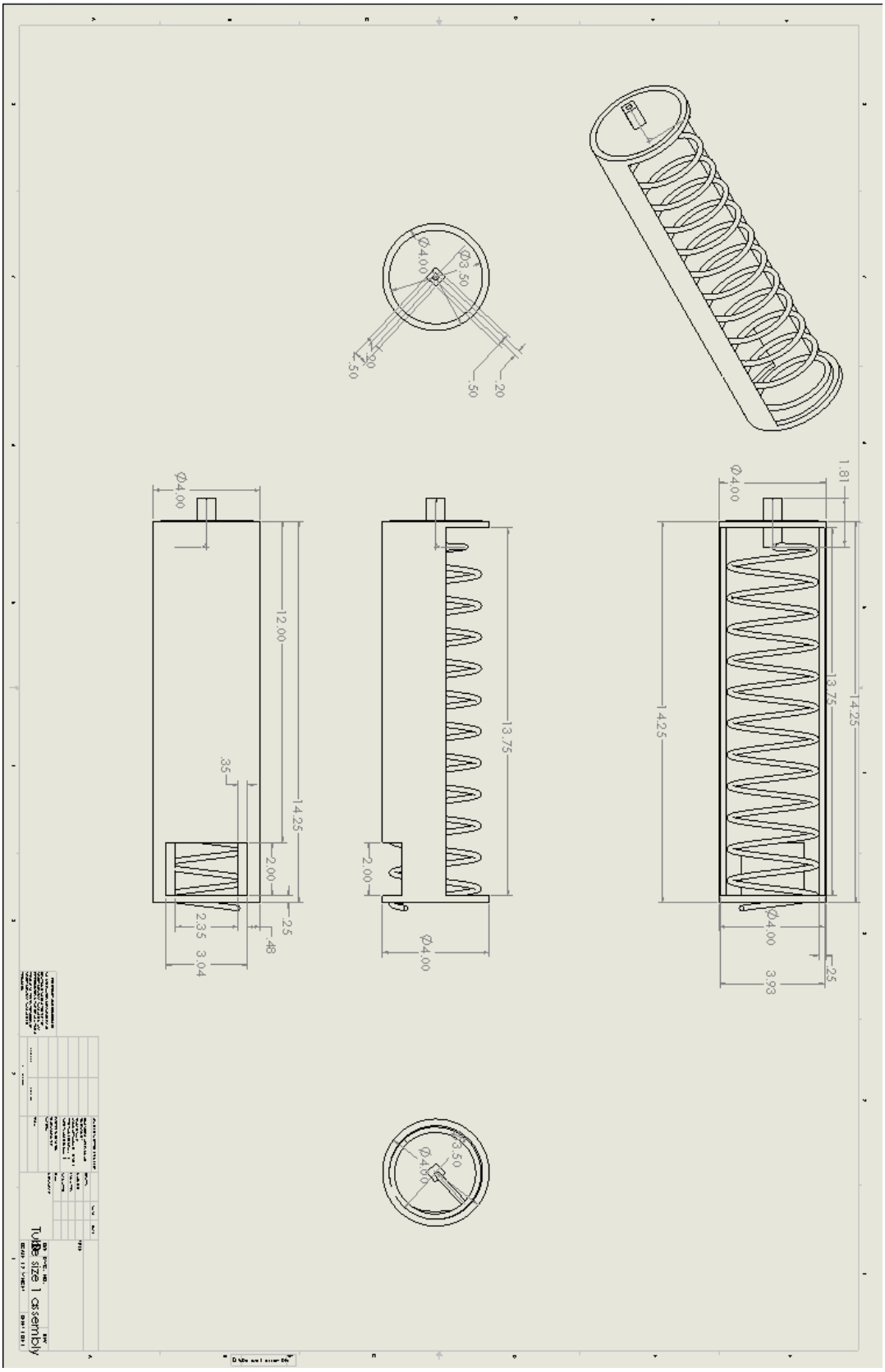

Figure 52: Size 1 Dispenser

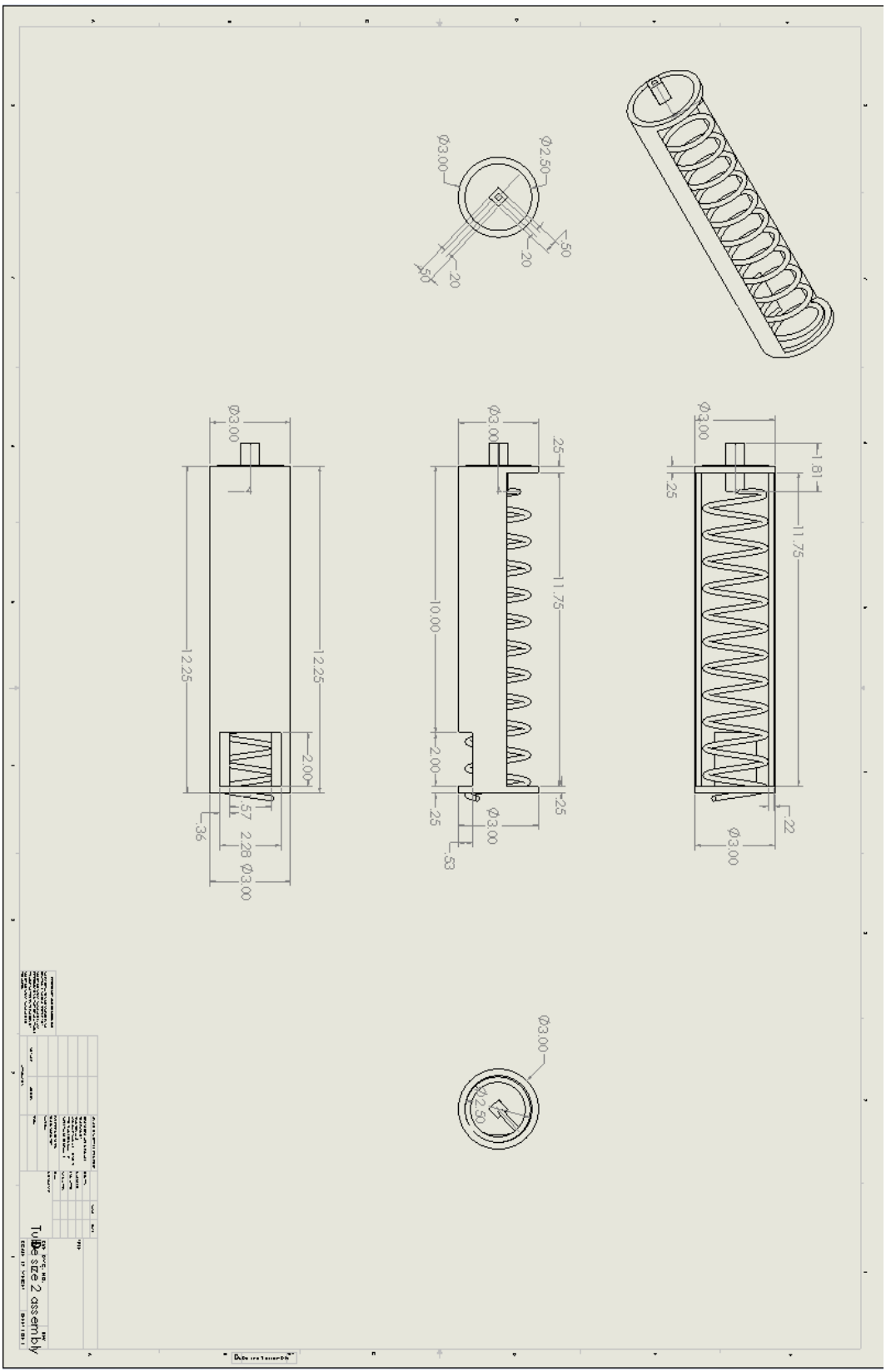

Figure 53: Size 2 Dispenser

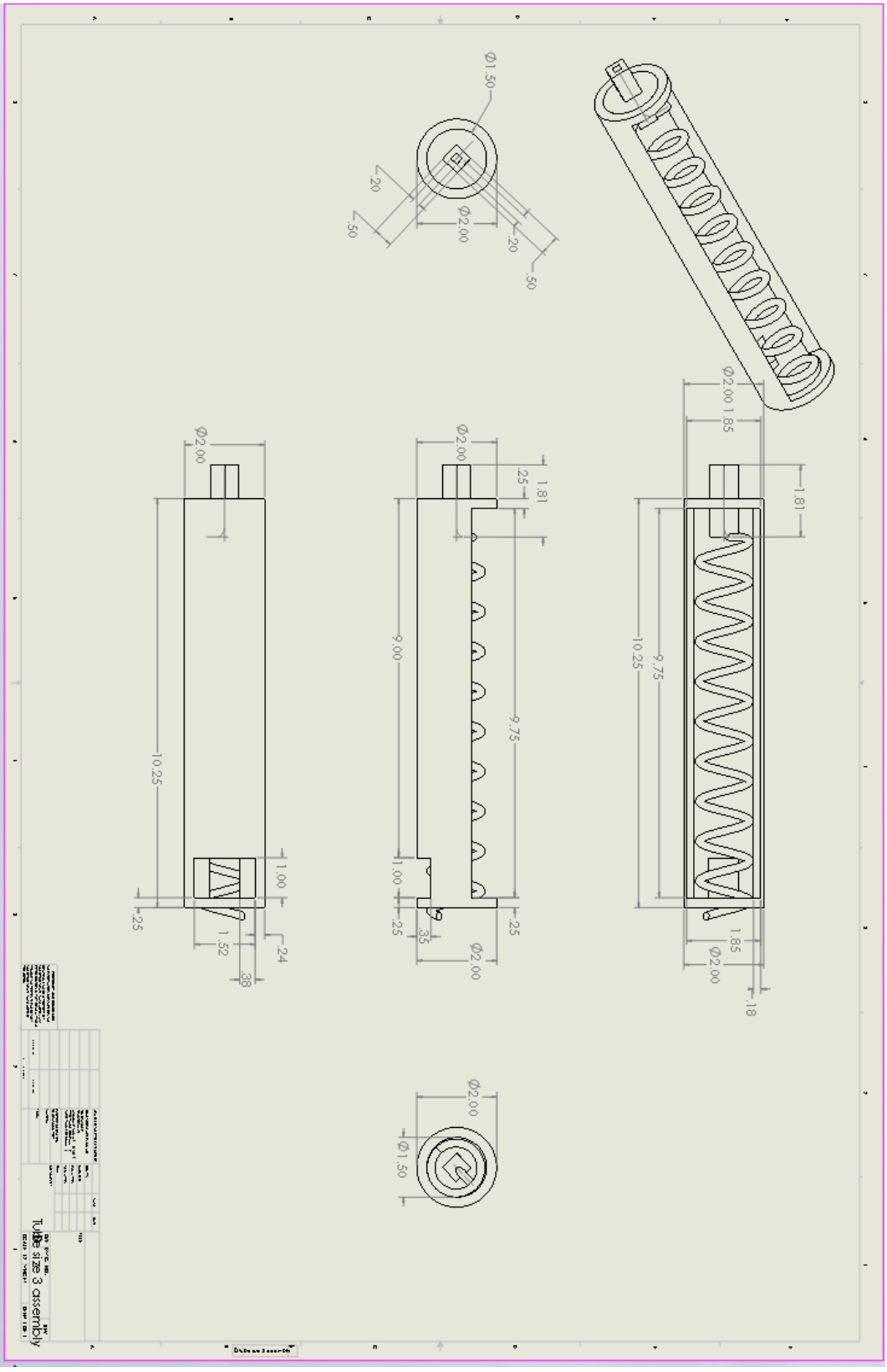

Figure 54: Size 3 Dispenser

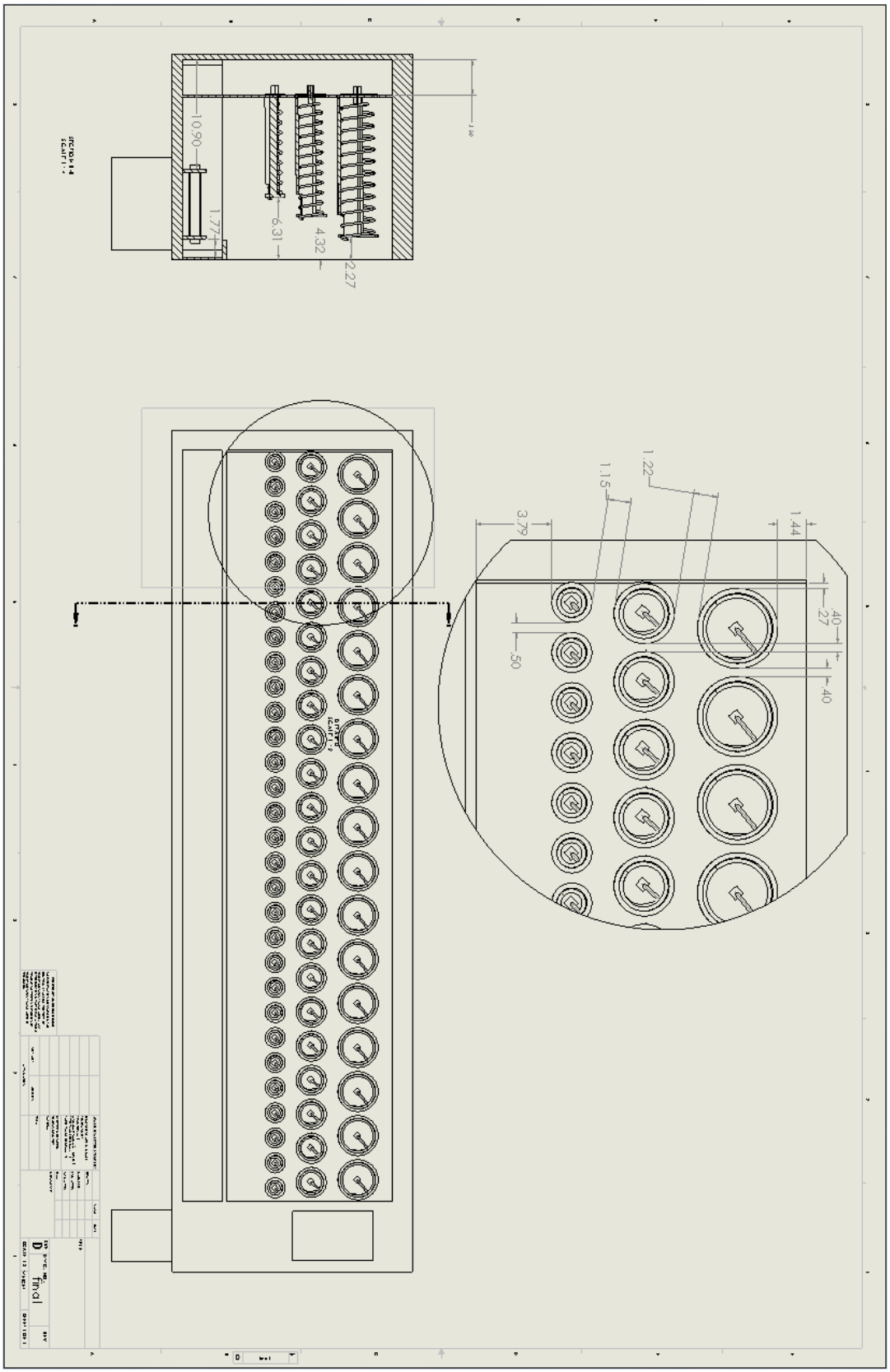

Figure 55: Final Design

Appendix G: Code

Overview

This Appendix covers the code run on the Arduino to test communication with the User Interface program, the current stable code for the user interface, and the unfinished code for adding cases into the data structure.

The code for the Arduino is a simple program that listens for a byte of data and after receiving one it activates the necessary pins waits a set amount of time then deactivates the pins. The current program is limited to handling one byte per tenth of a second.

The stable code (Version 1) has all of the features that we worked on except for sorting by case. These include:

- A logger that records user actions as well as some of the tasks during program initialization.
- A basic text file parser that allows the data about the pharmaceuticals to be changed without having to change the program itself.
- Ability to connect to the Arduino over a serial port (a USB).
- Sorting pharmaceuticals by category.

The code is broken up into these files:

- Com.java
- ComPort.java
- Index.java
- Item.java
- Tag.java
- TagUI.java
- TextFile.java
- VendingMachineGui1.java

The file system of the first version has two components the Index.txt and the pharmaceuticals. All of the files should reside in the same folder. The Index.txt is just a list of all the file names without the .txt added. The files for individual pharmaceuticals should have four parts

Layout of a Pharmaceutical text file:

- Name – to be shown on the GUI
- Byte – the byte value that relates to the position in the vending machine
    - This should be the same as the value the Arduino is expecting for the dispenser that contains the pharmaceutical.
- The list of tags – multi line
    - First the number of tags following this number
    - One tag per line
- The lines of the description – multi line
    - The number of lines that follow this number
    - One line of the description per line

The unfinished code (Version 2) is the current progress on adding cases to the GUI. This does not have all of the features from the first version implemented and the GUI functionality is not yet finished as sorting the pharmaceuticals into cases and categories has not yet been completed.

The code is broken up into these files:

- Case.java
- Index.java
- Item.java
- Tag.java
- TextFile.java
- VendingMachineGui3.java
- VMGUI.java

The file system is composed of the config.txt, one list for pharmaceuticals and another for cases, the pharmaceutical text files, and the case text files. The list for pharmaceuticals and the related text files are formatted the same as before. The list for case names is formatted like the list of pharmaceuticals.

The format for cases:

- Name – the name the GUI displays for this case
- Tags – multi line
    - Number of tags
    - One tag per line

- Items – multi line
    - Number of items
    - One item name per line
- Description – multi line
    - Number of lines
    - Lines of description

And the format for cases:

- Com port (ex. COM 3)
- Path to med index (ex. \med.text)
- Path to case index (ex. \case.text)

The directory where the information is stored is:

- Data(Folder)
    - Has the information need
- Log(Folder)
    - Destination of log files generated by the program
- Cases.txt
    - Points to cases in Data
- Config.txt
    - Points to meds and cases
- Meds.txt
    - Points to meds in Data

All of the text files should have no empty lines! In version 2's file system the cases and pharmaceuticals have their own sub folders.

Arduino code: VendingMachineDriver_v1.ino

```
/**
 *  Vending Machine Driver v1
 *  12/8/2013
 *  Author: William Jones
 */

/**
 *INFO::
 *  byte ranges for arduino serial monitor
 *    char(keyboard) = byte(used in arduino and returned to monitor)
 *    0-9 = 48-57
 *    a-z = 97-122
 *    A-Z = 65-90
 *
 *  to expand code to more pins (between 2 and 7(PWM)/13(Toggle))
 *    copy defines and change labels to the next pin
 *    copy setup for servo objects if needed
 *
 *  PWM functionality does not work
 *
 *
 *
 */

#include <TimerOne.h>
#include <Servo.h>

//config define's
```

```
//time constants
#define dispencerTime 10 // the time the dispenser should run time in ms (10ms or 1s)
#define beltTime 100// the time the conveyor belt should run
//control constants
#define conveyorPWMSignalDrive 180
#define conveyorPWMSignalStop 90
#define storagePWMSignalOpen 180
#define storagePWMSignalClose 0

// define pin2 (first free pin)
#define isConveyor true
#define pin2PWM false
#define serialOne 50 // compare with serial data to select this item

// copy this//
// define pin3
// used to set whether or not the Arduino should output pwm
#define pin3PWM false
// set what byte value this pin should activate for
#define serialTwo 51
// stop copying//

#define pin4PWM false
#define serialThree 52



// global varables
int incomingByte = 0;
```

```
// the count of the number of times the interupt has fired
// 0.1s
int timeCount = 0;
//currentTarget is the list of times to be watched
int currentTarget[10][2];
int currentTargetSize = 0;
// time that the conveyor should stop at in 0.1s
int conveyorEnd = 0;

// declare servo objects
  // max 8
Servo conveyorServo;
Servo pin2Servo;
Servo pin3Servo;
Servo pin4Servo;



// setup function a.k.a. things to do once
void setup()
{
  // initalize CurrentTarget array
  for(int i = 0; i < 10; i++){
    for(int j = 0; j < 2; j++){
      currentTarget[i][j] = 0;
    }
  }


  // Initialize the digital pins as an output.
```

```
pinMode(2, OUTPUT); // first free pin
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);

// setup servo objects
if(isConveyor && pin2PWM){// setup for conveyor
  Serial.println("Conveyor Servo Init");// prints to arduino serial monitor
  conveyorServo.attach(2);// attaches the servo object
}
if(!isConveyor && pin2PWM){// setup for dispenser
  pin2Servo.attach(2);
  Serial.println("Pin 2 Servo Init");
}

if(pin3PWM){// setup for dispenser
  pin3Servo.attach(3);
  Serial.println("Pin 3 Servo Init");
}

if(pin4PWM){
  pin4Servo.attach(4);
  Serial.println("Pin 4 Servo Init");
}



Serial.begin(115200);// set serial com baud rate

Timer1.initialize(100000); // set a timer of length 100000 microseconds (or 0.1 sec - or 10Hz)
Timer1.attachInterrupt( timerIsr ); // attach the service routine here
```

```
}

// main program
void loop()
{
  //Serial.println("Main Loop");
  if (Serial.available() > 0) {// if detects non zero byte
    // read the incoming byte:
    incomingByte = Serial.read();// get byte

    // say what you got:
    Serial.print("I received: ");//print to serial monitor
    Serial.println(incomingByte);
    setAction(incomingByte);// give byte to function setAction
    incomingByte = 0;// clear byte
  }
}



/**
 *  Timer ISR
 *
 *  Update timeCount
 */

void timerIsr()
{
  timeCount++;// increment count
```

```
if (timeCount == currentTarget[0][0]){//checks the head of the queue against the current time count
  /*
  Serial.print(timeCount);
  Serial.print(",");
  Serial.print(currentTarget[0][0]);*/
  Serial.println("Close Storage");


  stopAction(currentTarget[0][1]);// stops a dispenser
  nextTarget();// advances the queue


  /*
  boolean runLoop = true;
  while(runLoop){
    //Serial.println("Loop");
    stopAction(currentTarget[0][1]);
    if(currentTarget[0][0] != currentTarget[1][0]){
      runLoop = false;
    }
    nextTarget();
  }
  */
}
// check a second time  *does not function the way it should*
if (timeCount == currentTarget[0][0]){/*
  Serial.print(timeCount);
  Serial.print(",");
  Serial.print(currentTarget[0][0]);*/
```

```
    Serial.println("Close Storage 2");

    stopAction(currentTarget[0][1]);
    nextTarget();

  }

  // checks to see if the timecount is equal the when the conveyor should stop
  if (timeCount == conveyorEnd){
    Serial.println("Stop Conveyor");
    if(pin2PWM){// stop if PWM
      Serial.println("PWM");
      conveyorServo.write(conveyorPWMSignalStop);
    }
    else{// stop if high
      Serial.println("State");
      digitalWrite(2,LOW);
    }

  }
}

// takes a byte, checks if a pin should be activated, if there is a pin it is activated and added to the queue
void setAction(int aByte){ // takes a byte and starts actions and sets watch times for stopping
  int time = timeCount;// get current count
  Serial.print("Time:");
  Serial.println(time);
  //set conveyor or output on pin 2 // setup for first/conveyor output
  if(aByte != 0){// if byte is not zero
```

```
if(isConveyor){// if this microcontroller handles the conveyor
  Serial.print("Start Conveyor:");
  conveyorEnd = time + beltTime;// adds a fixed time to the conveyor
  if(pin2PWM){// start with pwm
    Serial.println("PWM");
    conveyorServo.write(conveyorPWMSignalDrive);
  }
  else{// start with High
    Serial.println("State");
    digitalWrite(2,HIGH);
  }
}
if(!isConveyor && aByte == serialOne){// if pin 2 is not the conveyor
  Serial.print("Open Storage 1(pin2:)");
  int endTime = time + dispencerTime;// adds a fixed time for a dispenser
  addWatch(endTime,2);
  if(pin2PWM){
    Serial.println("PWM");
    pin2Servo.write(storagePWMSignalOpen);
  }
  else{
    Serial.println("State");
    digitalWrite(2,HIGH);
  }
}
// the basic form for adding pins for controlling dispansers
//copy this//
if(aByte == serialTwo){
  Serial.print("Open Storage 2(pin3):");
```

```
  int endTime = time + dispencerTime;
  addWatch(endTime,3);
  if(pin3PWM){
    Serial.println("PWM");
    pin3Servo.write(storagePWMSignalOpen);
  }
  else{
    Serial.println("State");
    digitalWrite(3,HIGH);
  }
 }
 //stop copying//

 if(aByte == serialThree){
  Serial.print("Open Storage 3(pin4):");
  int endTime = time + dispencerTime;
  addWatch(endTime,4);
  if(pin4PWM){
    Serial.println("PWM");
    pin4Servo.write(storagePWMSignalOpen);
  }
  else{
    Serial.println("State");
    digitalWrite(4,HIGH);
  }
 }

}
```

```
}

// takes a given pin and tells it to stop a dispenser
void stopAction(int pin){
 if(pin == 2){
  Serial.print("Close Storage 1(pin2):");
  if(pin2PWM){
   Serial.println("PWM");
   pin2Servo.write(storagePWMSignalOpen);
  }
  else{
   Serial.println("State");
   digitalWrite(2,LOW);
  }
 }
 //for every pin used the program needs to know to stop it
 //copy this//
 if(pin == 3){
  Serial.print("Close Storage 2(pin3):");
  if(pin3PWM){
   Serial.println("PWM");
   pin3Servo.write(storagePWMSignalOpen);
  }
  else{
   Serial.println("State");
   digitalWrite(3,LOW);
  }
 }
 //stop copying//
```

```
  if(pin == 4){
    Serial.print("Close Storage 3(pin4):");
    if(pin4PWM){
      Serial.println("PWM");
      pin4Servo.write(storagePWMSignalOpen);
    }
    else{
      Serial.println("State");
      digitalWrite(4,LOW);
    }
  }
}

// adds an entry to the queue in the form of (time, pin)
void addWatch(int time, int pin){
  //Serial.println("add");
  int pos = currentTargetSize;// gets the next free space
  if (pos < 10){// if space is in the queue
    currentTarget[pos][0] = time;// set the time
    currentTarget[pos][1] = pin;// set the pin
    currentTargetSize++;// set next as first free space
  }

}

// advances the queue by removeing the first and moving everything up
void nextTarget(){ // advances the watch list
  Serial.println("next");
  for(int i = 0; i < 9; i++){// move everything up
```

```
  currentTarget[i][0] = currentTarget[i+1][0];
  currentTarget[i][1] = currentTarget[i+1][1];
 }
 //fills in the last space
 currentTarget[9][0] = 0;
 currentTarget[9][1] = 0;
 if(currentTargetSize > 0)
  currentTargetSize--;

 /*
 currentTarget[0] = currentTarget[1];
 currentTarget[1] = currentTarget[2];
 currentTarget[2] = currentTarget[3];
 currentTarget[3] = currentTarget[4];
 currentTarget[4] = currentTarget[5];
 currentTarget[5] = currentTarget[6];
 currentTarget[6] = currentTarget[7];
 currentTarget[7] = currentTarget[8];
 currentTarget[8] = currentTarget[9];
 currentTarget[9] = 0;
 currentTargetSize--;
 */
}
```

Version 1: Com.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 * This code does not belong to us.
 * Some changes to this code were made by us.
 *
 */
import java.util.LinkedList;

import gnu.io.CommPortIdentifier;


public class Com {

	public Com(){

	}

	/**
	 * rxtx example code
	 *
	 * @param portType
	 * @return
	 */
    public void listPorts()
    {
        @SuppressWarnings("unchecked")
             java.util.Enumeration<CommPortIdentifier> portEnum =
CommPortIdentifier.getPortIdentifiers();
        while ( portEnum.hasMoreElements() )
        {
            CommPortIdentifier portIdentifier = portEnum.nextElement();
            System.out.println(portIdentifier.getName()  +  " - " +
getPortTypeName(portIdentifier.getPortType()) );
        }
    }

    static String getPortTypeName ( int portType )
    {
        switch ( portType )
        {
            case CommPortIdentifier.PORT_I2C:
                return "I2C";
            case CommPortIdentifier.PORT_PARALLEL:
                return "Parallel";
            case CommPortIdentifier.PORT_RAW:
                return "Raw";
            case CommPortIdentifier.PORT_RS485:
                return "RS485";
            case CommPortIdentifier.PORT_SERIAL:
                return "Serial";
            default:
                return "unknown type";
        }
```

```
    }

    public LinkedList<String> listOfPorts(){
       LinkedList<String> portList = new LinkedList<String>();

       @SuppressWarnings("unchecked")
             java.util.Enumeration<CommPortIdentifier> portEnum =
CommPortIdentifier.getPortIdentifiers();
        while ( portEnum.hasMoreElements() )
        {
            CommPortIdentifier portIdentifier = portEnum.nextElement();
            portList.add(portIdentifier.getName());
        }

       return portList;
    }
}
```

Version 1: ComPort.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 * This code does not belong to us.
 * Some changes to this code were made by us.
 *
 */
import java.io.InputStream;
import java.io.OutputStream;

import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;


public class ComPort {

	CommPort commPort;
	String type;
	InputStream in;
	OutputStream out;

	public ComPort(String portName){
		try{
		  CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);
		 if ( portIdentifier.isCurrentlyOwned() )
		 {
		     System.out.println("Error: Port is currently in use");
		 }
		 else
		 {

		     CommPort commPort =
portIdentifier.open(this.getClass().getName(),2000);

		     if ( commPort instanceof SerialPort )
		     {
		         SerialPort serialPort = (SerialPort) commPort;

//serialPort.setSerialPortParams(57600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,Se
rialPort.PARITY_NONE);

serialPort.setSerialPortParams(115200,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,Ser
ialPort.PARITY_NONE);

		         this.commPort = serialPort;
		         this.type = "Serial";
		         this.in = serialPort.getInputStream();
		         this.out = serialPort.getOutputStream();

		         //(new Thread(new SerialReader(in))).start();
		         //(new Thread(new SerialWriter(out))).start();
```

```
                    System.out.println("Connected - " + portName);


                }
                else
                {
                    System.out.println("Error: Only serial ports are handled by
this example.");

                }


              }
            }
            catch(Exception e){
                   System.out.println("Error:Unable to connect to " + portName);
            }

      }

}
```

Version 1: Index.java

```
/**

 * This code is used for the Ambulance Interior Storage Optimization IPQ.

 *

 */


import java.util.LinkedList;

import java.util.logging.Logger;


/**

 * This class is used to store data for the gui to use as well as import the data and process it

 * @author William Jones

 */

public class Index {

        static Logger logger = Logger.getLogger("Vending Machine");// connect to the logger

   LinkedList<ComPort> ports;

   LinkedList<Item> items;


   LinkedList<String> tagList;

   LinkedList<Tag> tagIndex;


   /**

    * This is the constructor used to create the index

    * @param itemList a textfile with the locations of the pharmacutical info files

    * @param ports the ports that are connected to this program

    */
```

```
public Index(TextFile itemList, LinkedList<ComPort> ports){
   this.ports = ports;// store the ports
   this.items = getItems(itemList);// import the items
   //this.items = initTest();

   this.tagList = this.getTags(this.items);// extract the tags as strings
   this.tagIndex = this.sortTag(this.items, this.tagList);// create the tag objects and link the items

}

/**
 * This function is used to get the list of tags as strings form the given collection of items
 * @param items the list if items to sort
 * @return the list of tag names
 */
LinkedList<String> getTags(LinkedList<Item> items){
   LinkedList<String> tags = new LinkedList<>();// initialize return list
   int numberOfItems = items.size();// get the number of items to search through
   for(int i = 0; i < numberOfItems; i++){// iterate through the items
      Item current = items.get(i);// get the current item
      LinkedList<String> newTags = current.tags;// get the tags from the current item
      for(int j = 0; j < newTags.size(); j++){// iterate through the tags
         if(!tags.contains(newTags.get(j))){// if the tag is new add the tag
            tags.addLast(newTags.get(j));
         }
```

```
        }
    }

    return tags;// return the list of tags
}

/**
 * This function links items with the tags that apply to them
 * @param items list of all items
 * @param tagList list of discovered tags
 * @return a list of tag objects
 */
LinkedList<Tag> sortTag(LinkedList<Item> items, LinkedList<String> tagList){
    LinkedList<Tag> tags = new LinkedList<>();// initalize
    for(int i = 0; i < tagList.size(); i++){// for each tag
        String currentTag = tagList.get(i);// get the current tag
        //System.out.println(currentTag);
        Tag tag = new Tag(currentTag);// create a new tag object
        for(int j = 0; j < items.size(); j++){// for each item
            if(items.get(j).tags.contains(currentTag)){// does the item have this tag
                tag.add(items.get(j));// if it does add this item to this tag object
            }
        }
        tags.addLast(tag);// add this tag object to the list
    }
```

```
    return tags;// return the list of tag objects
}

/**
 * This function was used to simulate data before importing from files was available
 * @return
 */
public static LinkedList<Item> initTest(){
    LinkedList<Item> list = new LinkedList<>();

    //ToDo create items
    Item item1 = new Item("Band-Aid", 51);
    item1.addTag("Bandage");
    item1.addInfo("Used to treat small cuts");
    list.addLast(item1);

    Item item2 = new Item("Ace Bandage", 52);
    item2.addTag("Bandage");
    item2.addTag("Wrap");
    list.addLast(item2);

    Item item3 = new Item("Advil", 53);
    item3.addTag("Pill");
    list.add(item3);
```

```
    return list;
}

/**
 * This function uses the item list in index.txt to import the item data from files
 * @param itemList the data from index.txt
 * @return the list of item objects
 */
public static LinkedList<Item> getItems(TextFile itemList){
    LinkedList<Item> list = new LinkedList<>();// initialize
    while(itemList.content.size() > 0){// while there are more items
        String item = itemList.content.pollFirst();// get and remove the first in the list
        // create the path to the data of the item and import the file
        TextFile itemFile = new TextFile("C:\\Program Files\\VendingMachine\\" + item + ".txt");
        logger.info("Add Item - " + item);// print logger
        System.out.println("File:" + item);// print to console
        for(int i = 0; i < itemFile.content.size();i++){// print the file to the console
                System.out.println(i + ":" + itemFile.content.get(i));
        }
        String itemName = itemFile.content.get(0);// get the name of the pharmaceutical
        int value = new Integer(itemFile.content.get(1));// get the byte value
        Item item1 = new Item(itemName, value);// create the item
        int numberOfTags = new Integer(itemFile.content.get(2));// get the number of tags
        if(numberOfTags > 0){// get the tags
```

```
                System.out.println("Tags");
                for(int i = 1; i <= numberOfTags; i++){
                        String line = itemFile.content.get(2 + i);
                        item1.addTag(line);
                        System.out.println(line);
                }
        }
        // get the number of lines in the description
        int numberOfDescription = new Integer(itemFile.content.get(3 + numberOfTags));
        if(numberOfDescription > 0){// get the description
                int offset = 3 + numberOfTags;
                System.out.println("Info");
                for(int i = 1; i <= numberOfDescription; i++){
                        String line = itemFile.content.get(offset + i);
                        item1.addInfo(line);
                        System.out.println(line);
                }
        }
        //item1.addTag("Bandage");
        list.addLast(item1);// add the item
    }

    return list;// return the list of items
  }
}
```

Version 1: Item.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */
import java.util.LinkedList;

/**
 * This class is used to Store information about a pharmaceutical
 * @author William Jones
 */
public class Item {
    String name;
    int id;
    LinkedList<String> tags;
    LinkedList<String> info;

    /**
     * This is the basic constructor for this class
     * @param name the name of the item being created
     * @param id the byte value that gets sent to the arduino
     */
    public Item(String name, int id){
        this.name = name;
        this.id = id;
        this.tags = new LinkedList<>();
        this.info = new LinkedList<>();
    }

    /**
     * This is a constructor that includes a list of tags for the item.
     * @param name item name
     * @param id byte value
     * @param tags list of tags
     */
    public Item(String name, int id, LinkedList<String> tags){
        this.name = name;
        this.id = id;
        this.tags = tags;
        this.info = new LinkedList<>();
    }

    /**
     * This constructor accepts all fields that the item stores
     * @param name the item name
     * @param id the byte value
     * @param tags the list of tags
     * @param info the description as a list of strings
     */
    public Item(String name, int id, LinkedList<String> tags, LinkedList<String>
info){
        this.name = name;
        this.id = id;
        this.tags = tags;
```

```
        this.info = info;
    }

    /**
     * This function is used to rename an item if need be.
     * @param name the new name
     */
    public void rename(String name){
        this.name = name;
    }

    /**
     * This function clears the tags that this item has
     */
    public void clearTags(){
        this.tags = new LinkedList<>();
    }

    /**
     * This function adds a new tag to this item
     * @param tag the tag to add
     */
    public void addTag(String tag){
        this.tags.addLast(tag);
    }

    /**
     * Checks to see if this item has the given tag
     * @param tag the tag to check for
     * @return
     */
    public boolean hasTag(String tag){
        return this.tags.contains(tag);
    }

    /**
     * This function clears the items description
     */
    public void clearInfo(){
        this.info = new LinkedList<>();
    }

    /**
     * This function adds a new line to the description
     * @param info a string to display
     */
    public void addInfo(String info){
        this.info.addLast(info);
    }
}
```

Version 1: Tag.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */
import java.util.LinkedList;

/**
 * This class is used to create a Tag object that has a list of all the items with
the given tag
 * @author William Jones
 */
public class Tag {

    String tag;
    LinkedList<Item> items;

    /**
     * Constructor for the tag object
     * @param tag a string
     */
    public Tag(String tag){
        this.tag = tag;
        this.items = new LinkedList<Item>();
    }

    /**
     * Function adds an item to the tag
     * @param item item object to be added
     */
    public void add(Item item){
        this.items.addLast(item);
    }

    /**
     * Returns a list of item names
     * @return linkedList of items
     */
    public LinkedList<String> listItems(){
        LinkedList<String> items = new LinkedList<>();
        for(int i = 0; i < this.items.size(); i++){// Iterate through the items
            items.add(this.items.get(i).name);
        }
        return items;
    }
}
```

Version 1: TagUI.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */
import java.util.LinkedList;
import java.util.logging.Logger;


import javax.swing.DefaultListModel;
import javax.swing.JList;
import javax.swing.ListSelectionModel;


/**
 * This is the User Interface
 * @author William Jones
 */
@SuppressWarnings("serial")
public class TagUI extends javax.swing.JFrame {
        static Logger logger = Logger.getLogger("Vending Machine");
   Index index;
   String[] tags;
   String[] medList;
   boolean tagSelected;
   boolean medSelected;
   Item selected;
```

```
    /**
     * Creates new Gui
     */
    public TagUI(Index index) {
        this.index = index;// keeps the index as part if this class

        this.tags = index.tagList.toArray(new String[0]);// gets the names of all the tags and converts the linked list to an array
        this.medList = new String[0];// initializes the second JList with an empty array
        this.tagSelected = false;// mark that no tag has been selected
        this.medSelected = false;// mark that no medicine/pharmaceutical has been selected
        this.selected = new Item("No Item", 0);// initializes the selected item with dummy data
        System.out.println("Setup Complete");
        initComponents();// initializes the gui components
        /*
        while(true){
            byte b = 0;
            try{
                    b = (byte)System.in.read();
                    System.out.println(b);
            }
            catch(Exception e){

            }

            if(b == (byte)'a'){
```

```
                System.out.println();
        }


    }
    //*/
}


/**
 * This function initializes the gui components
 */
@SuppressWarnings({ "unchecked", "rawtypes" })
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

        // initializes the components
    jScrollPane1 = new javax.swing.JScrollPane();
    jList1 = new javax.swing.JList();
    jButton1 = new javax.swing.JButton();
    jScrollPane2 = new javax.swing.JScrollPane();
    //jList2 = new javax.swing.JList();
    jScrollPane3 = new javax.swing.JScrollPane();
    jTextPane1 = new javax.swing.JTextPane();
    jScrollPane4 = new javax.swing.JScrollPane();
    jTextPane2 = new javax.swing.JTextPane();
```

```
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);// set the program to close if the window is closed


    jList1.setModel(new javax.swing.AbstractListModel() {// sets the data for the JList for tags
        String[] strings = tags;
        public int getSize() { return strings.length; }
        public Object getElementAt(int i) { return strings[i]; }
    });
    jList1.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);// sets the JList as a single selection
    jList1.addMouseListener(new java.awt.event.MouseAdapter() {// listens for the mouse
        public void mouseClicked(java.awt.event.MouseEvent evt) {// if mouse is clicked
            jList1MouseClicked(evt);// do this
        }
    });
    jScrollPane1.setViewportView(jList1);// put the JList inside the scroll panel


    jButton1.setText("Request");// labels the button
    jButton1.addActionListener(new java.awt.event.ActionListener() {// listens for button press
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);// do this on button press
        }
    });


    /*jList2.setModel(new javax.swing.AbstractListModel() {
        String[] strings = medList;
```

```
        public int getSize() { return strings.length; }
        public Object getElementAt(int i) { return strings[i]; }
    });//*/
    jList2.addMouseListener(new java.awt.event.MouseAdapter() {// listens for the mouse
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jList2MouseClicked(evt);// if a medicine is clicked do this
        }
    });
    jScrollPane2.setViewportView(jList2);// put the JList inside the scroll panel

    jScrollPane3.setViewportView(jTextPane1);// put the textpane inside this scroll panel

    jScrollPane4.setViewportView(jTextPane2);

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());// layout the components
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 80, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 80, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addGroup(layout.createSequentialGroup()
                  .addGap(0, 135, Short.MAX_VALUE)
                  .addComponent(jButton1))
               .addComponent(jScrollPane3)
               .addComponent(jScrollPane4))
            .addContainerGap())
      );
      layout.setVerticalGroup(
         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
               .addGroup(layout.createSequentialGroup()
                  .addComponent(jScrollPane3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                  .addComponent(jScrollPane4)
                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                  .addComponent(jButton1))
               .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 278,
Short.MAX_VALUE)
               .addComponent(jScrollPane2))
            .addContainerGap())
      );
```

```
    pack();




}// </editor-fold>

/**
 * When the mouse is clicked on the tag JList
 * @param evt
 */
@SuppressWarnings("unchecked")
        private void jList1MouseClicked(java.awt.event.MouseEvent evt) {

    //System.out.println("click");
    String selection1 = (String)jList1.getSelectedValue();// get the selected tag
    //System.out.println(selection1);
    for(int i = 0; i < index.tagIndex.size(); i++){// iterate through the list of tags
         //System.out.println(i + ":" + selection1 + "=" + index.tagIndex.get(i).tag);
       if(index.tagIndex.get(i).tag == selection1){// if this tag is the one selected
          //String[] newData = index.tagIndex.get(i).listItems().toArray(new String[0]);
          //this.medList = index.tagIndex.get(i).listItems().toArray(new String[0]);
          //jList2.setListData(index.tagIndex.get(i).listItems().toArray(new String[0]));
          //jList2.setListData(newData);
         //DefaultListModel model = new DefaultListModel();
```

```
            listModel2.removeAllElements();// clear the medicine JList
            LinkedList<String> items = index.tagIndex.get(i).listItems();// gets the list of items
            while(items.size() > 0){//while there are items in the copied list
                    //System.out.println(items.size() + items.peek());
                    //model.addElement(items.poll());
                    listModel2.addElement(items.poll());// get and remove the item and add it to the list
model
            }
            //this.listModel2  = model;
            //System.out.println("Found");
          }
        }
        logger.info("User Selected Tag - " + selection1);// log the change in tag
        //jList2 = new JList(model);
        this.tagSelected = true;// mark that the tag is selected
        this.medSelected = false;// mark that med needs to be selected
        this.selected = new Item("No Item", 0);// clear the selection
        jTextPane1.setText("");// clear the name field
        jTextPane2.setText("");// clear the description field
    }

    /**
     * On button press
     * @param evt
     */
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        logger.info("User Requested - " + selected.name);// log the request
    try{
       //byte b = 51;
       index.ports.getFirst().out.write(selected.id);// sent byte value to the arduino
    }
    catch(Exception e){


    }
}

/**
 * This is called when the med JList is clicked
 * @param evt
 */
private void jList2MouseClicked(java.awt.event.MouseEvent evt) {


        String selection2 = (String)jList2.getSelectedValue();// gets the selected med
        System.out.println(selection2);
        for(int i = 0; i < index.items.size(); i++){//for the items in the tag
                if(index.items.get(i).name == selection2){// check for match
                        selected = index.items.get(i);// if match set as selected
                }
        }
        logger.info("User Selected - " + selection2);// log the selection
```

```
        this.medSelected = true;// mark that med is selected
        jTextPane1.setText(selection2);// set the name
        String text = "";// initialize description
        for(int i = 0; i < selected.info.size(); i++){
                text = text + selected.info.get(i)+ "\n";// add line
        }
        jTextPane2.setText(text);// set description
        //jTextPane2.setText(desc.toString());
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
        @SuppressWarnings("rawtypes")
        private javax.swing.JList jList1;
@SuppressWarnings("rawtypes")
        DefaultListModel listModel2 = new DefaultListModel();
@SuppressWarnings({ "rawtypes", "unchecked" })
        private javax.swing.JList jList2 = new JList(listModel2);
//private javax.swing.JList jList2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JScrollPane jScrollPane3;
private javax.swing.JScrollPane jScrollPane4;
private javax.swing.JTextPane jTextPane1;
private javax.swing.JTextPane jTextPane2;
```

```
    // End of variables declaration


    /**
     * Update the list model for med JList
     * @param tag the selected tag
     * @return the JList model
     */
    @SuppressWarnings({ "unchecked", "rawtypes", "unused" })
            private DefaultListModel updateList(Tag tag){
            DefaultListModel model = new DefaultListModel();
            LinkedList<String> items = tag.listItems();
            while(items.size() > 0){
                    model.addElement(items.poll());
            }
            return model;
    }



}
```

Version 1: TextFile.java

```
/**
 * This code is part of the Ambulance Interior Storage Optimization IPQ
 *
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.LinkedList;

/**
 * This class is used to read in a text file and store the lines as a linked list
 * @author William Jones
 *
 */
public class TextFile {

	LinkedList<String> content;

	/**
	 * The constructor for this class, it accepts a string with the path to the
text file.
	 * @param fileName
	 */
	TextFile(String fileName){
		content = new LinkedList<String>();// initialize linkedList
		try{//try to read the file
			BufferedReader in = new BufferedReader(new FileReader(fileName));
			String text;
			while (in.ready()) {
				text = in.readLine();// get line
				content.addLast(text);// add line to the end of the list
				System.out.println(text);// print the line to the
console
			}

			in.close();// close the file reader
		}
		catch(Exception e){

		}
	}

}
```

Version 1: VendingMachineGui1.java

```
/**
 * This code is part of the Ambulance Interior Storage Optimization IPQ
 *
 */

// imports
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.LinkedList;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;

/**
 * This is the main class of this project and is responsible for setting up the
program
 * @author William Jones
 */
public class VendingMachineGui1 {

    /**
     * This is the main function that is called to start the program
     * @param args the command line arguments
     */
    public static void main(String[] args) {
      Date date = new Date();// Initialize date field
      SimpleDateFormat ft = new SimpleDateFormat("MM-dd-yyyy_hh-mm-ss");// set
format
        final Logger logger = Logger.getLogger("Vending Machine");// create logger
for the program
        FileHandler fh;// declare the file handler for the logger
        System.out.println(ft.format(date));// prints date to console

        try {

            // This block configure the logger with handler and formatter
             // creates a new log file in C:\Program Files\VendingMachin\Log\
            fh = new FileHandler("C:\\Program Files\\VendingMachine\\Log\\Log_" +
ft.format(date) + ".log");
            logger.addHandler(fh);// passes the file to the logger
            SimpleFormatter formatter = new SimpleFormatter();// sets the format for
the file
            fh.setFormatter(formatter);

            // the following statement is used to log any messages
            logger.info("Start of Log");

        } catch (SecurityException e) {// exceptions
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        System.out.println("Start: File Test");
        logger.info("Read Index File");
        // get the text file at C:\Program Files\VendingMachine\
         TextFile itemIndex = new TextFile("C:\\Program
Files\\VendingMachine\\Index.txt");

        // Gui formating
                try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(TagUI.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(TagUI.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(TagUI.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(TagUI.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        }
        //*/
        //Test All ComPorts
        System.out.println("File Test: Com Ports");
        Com ports = new Com();
             ports.listPorts();
             LinkedList<String> activePorts = ports.listOfPorts();// list of all com
ports

             LinkedList<ComPort> comPorts = new LinkedList<>();// initialize the list
of used com ports

             /* input stream
             if(portActive(activePorts,"COM3")){
                   try{
                         comPorts.add(new ComPort("COM3"));
                   }
                   catch(Exception e){
                   }
             }
             */
```

```
            if(portActive(activePorts,"COM3")){// check if COM3 is available (Com
port may change on different computers
                    try{
                           comPorts.add(new ComPort("COM3"));// if it is add it to
the list
                    }
                    catch(Exception e){
                    }
             }

                if(comPorts.size() >= 0){// if there is a com port create index of
data
                    System.out.println("File Test: Make Index");
                    final Index index = new Index(itemIndex, comPorts);// creates
index
                    //new Gui(comPorts.getFirst());

                    //TagUI gui = new TagUI(index);

                    java.awt.EventQueue.invokeLater(new Runnable() {//start GUI
            public void run() {
             logger.info("Start GUI");
                new TagUI(index).setVisible(true);
            }
        });
                }
    }

    /**
     * This function checks if the given COM port is available
     * @param ports list of all ports
     * @param port port to check
     * @return boolean
     */
    public static boolean portActive(LinkedList<String> ports, String port){
        return ports.contains(port);
    }


}
```

Version 1: Index.txt

Version 2: Case.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */
package vendingmachinegui3;

import java.util.LinkedList;

/**
 * This class is used for making case objects to link relevant items
 * @author Will
 *
 */
public class Case {
	String name;
	LinkedList<String> tags;
	LinkedList<Item> items;
	LinkedList<String> description;
	boolean getAll;

	/**
	 * This is the constructor used to create a case for all items in a category
	 * @param name the name of the case
	 * @param itemList the list of items the case contains
	 */
	public Case(String name, LinkedList<Item> itemList){
		this.name = name;
		this.tags = new LinkedList<String>();
		this.items = itemList;
		this.description = new LinkedList<String>();
		this.description.add("This is the list of all the medicines in the
category.");
		this.getAll = false;// block getting all items
	}

	/**
	 * This constructor is used to create a case with a case text file and the
list of all items
	 * @param file the file describing the case
	 * @param itemList the list of items to search through
	 */
	public Case(TextFile file, LinkedList<Item> itemList){
		this.name = file.content.get(0);
		this.tags = new LinkedList<String>();
		// find tags
		int numberOfTags = new Integer(file.content.get(1));
		for(int i = 1; i <= numberOfTags; i++){
			String line = file.content.get(1 + i);
			this.tags.add(line);
			System.out.println(line);
		}
		// find items
		int numberOfItems = new Integer(file.content.get(2 + numberOfTags));
```

```
		for(int i = 1; i <= numberOfItems; i++){
			String line = file.content.get(2 + numberOfTags + i);
			for(int j = 0; j < itemList.size(); j++){
				if(line == itemList.get(j).name){
					this.items.addLast(itemList.get(j));
				}
			}
			System.out.println(line);
		}
		// find description
		int sizeOfDescription = new Integer(file.content.get(3 + numberOfTags +
numberOfItems));
		for(int i = 1; i <= sizeOfDescription; i++){
			String line = file.content.get(3 + numberOfTags + numberOfItems +
i);
			this.description.add(line);
			System.out.println(line);
		}
		// enable get all items
		this.getAll = true;
	}

}
```

Version 2: Index.java

```
package vendingmachinegui3;

/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */


import java.util.LinkedList;
import java.util.logging.Logger;


/**
 * This is the class for indexing the data
 * @author William Jones
 */
public class Index {
	// connect to the logger
	static Logger logger = Logger.getLogger("Vending Machine");
	String path;

	LinkedList<Item> items;
	LinkedList<Case> cases;
	LinkedList<String> tagNames;
	LinkedList<Tag> tags;

	/**
	 * This is the constructor for the index
	 * @param path the path to search for data files
	 * @param config this is the text file that gives info that the program needs
	 */
```

```
public Index(String path, TextFile config){
        this.path = path;
        // this is the text file for the item index
        TextFile itemList = new TextFile(path + config.content.get(1));
        // import items
        this.items = getItems(path, itemList);
        // this is the text file for the case index
        TextFile caseList = new TextFile(path + config.content.get(2));
        // import cases
        this.cases = getCases(path, caseList, this.items);
        // find tags
        this.tagNames = findTags();
        // create tag objects
        this.tags = generateTags();



}

/**
 * This function takes a path and the item index text file and
 * imports the items into the program
 * @param path the location to look for the item files
 * @param itemList the item index text file
 * @return list of items
 */
public LinkedList<Item> getItems(String path, TextFile itemList){
   LinkedList<Item> list = new LinkedList<>();
   while(itemList.content.size() > 0){
        String item = itemList.content.pollFirst();
```

```
TextFile itemFile = new TextFile(path + "\\data\\" + item + ".txt");
logger.info("Add Item - " + item);
System.out.println("File:" + item);
/*
 * This try/catch is used to let the program run even if the item is not defined
 * in the file system
 */
try{
	for(int i = 0; i < itemFile.content.size();i++){
		System.out.println(i + ":" + itemFile.content.get(i));
	}
	String itemName = itemFile.content.get(0);
	int value = new Integer(itemFile.content.get(1));
	Item item1 = new Item(itemName, value);
	int numberOfTags = new Integer(itemFile.content.get(2));
	if(numberOfTags > 0){
		System.out.println("Tags");
		for(int i = 1; i <= numberOfTags; i++){
			String line = itemFile.content.get(2 + i);
			item1.addTag(line);
			System.out.println(line);
		}
	}
	int numberOfDescription = new Integer(itemFile.content.get(3 + numberOfTags));
	if(numberOfDescription > 0){
		int offset = 3 + numberOfTags;
		System.out.println("Info");
		for(int i = 1; i <= numberOfDescription; i++){
			String line = itemFile.content.get(offset + i);
```

```
                                item1.addInfo(line);
                                System.out.println(line);
                        }
                }
                //item1.addTag("Bandage");
                list.addLast(item1);
        }
        catch(Exception e){
                logger.warning("Missing item");
        }
   }



   return list;
}

/**
 * This function is used to get the cases for the program
 * @param path the location of the text files
 * @param caseList the index of cases text files
 * @param items the items that have been imported
 * @return the case objects that are created
 */
public LinkedList<Case> getCases(String path,TextFile caseList, LinkedList<Item> items){
        LinkedList<Case> list = new LinkedList<Case>();
        for(int i = 0; i < caseList.content.size(); i++){
                try{
                        TextFile caseFile = new TextFile(path + "\\Data\\" + caseList.content.get(i));
```

```
                    Case newCase = new Case(caseFile, items);
                    list.add(newCase);
                }
                catch(Exception e){
                    logger.warning("Missing Case");
                }
        }

        return list;
}

/**
 * This function is used to identify the tags of the cases and items
 * @return list of tags as a string
 */
LinkedList<String> findTags(){
        LinkedList<String> list = new LinkedList<String>();

        for(int i = 0; i < this.items.size(); i++){
                Item current = this.items.get(i);
                for(int j = 0; j < current.tags.size(); j++){
                        if(!list.contains(current.tags.get(j))){
                                list.add(current.tags.get(j));
                        }
                }
        }

        for(int i = 0; i < this.cases.size(); i++){
                for(int j = 0; j < this.cases.get(i).tags.size(); j++){
```

```
                        if(!list.contains(this.cases.get(i).tags.get(j))){
                                list.add(this.cases.get(i).tags.get(j));
                        }
                }
        }


        return list;
}

/**
 * This function uses the list of tags stored in this object and creates the tag objects
 * @return
 */
/*
 * this function is not yet working and does not correctly create cases
 * for the gui to display
 */
LinkedList<Tag> generateTags(){
        LinkedList<Tag> list = new LinkedList<Tag>();
        for(int i = 0; i < this.tagNames.size(); i++){
                Tag current = new Tag(this.tagNames.get(i));
                for(int j = 0; j < this.cases.size(); j++){
                        if(this.cases.get(j).tags.contains(this.tagNames.get(i))){
                                current.cases.add(this.cases.get(j));
                        }
                }
                LinkedList<Item> taggedItems = new LinkedList<Item>();
                for(int k = 0; k < this.items.size(); k++){
```

```
                        if(this.items.get(k).tags.contains(this.tagNames.get(i))){
                                taggedItems.add(this.items.get(k));
                        }
                }
                //add case for all items in category
                current.cases.addFirst(new Case("All Items", taggedItems));
        }
        //add category for everything


        return list;
    }



}
```

Version 2: Item.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */

package vendingmachinegui3;

import java.util.LinkedList;

/**
 * This class is used to Store information about a pharmaceutical
 * @author William Jones
 */
public class Item {
    String name;
    int id;
    LinkedList<String> tags;
    LinkedList<String> info;

    /**
     * This is the basic constructor for this class
     * @param name the name of the item being created
     * @param id the byte value that gets sent to the arduino
     */
    public Item(String name, int id){
        this.name = name;
        this.id = id;
        this.tags = new LinkedList<>();
        this.info = new LinkedList<>();
    }

    /**
     * This is a constructor that includes a list of tags for the item.
     * @param name item name
     * @param id byte value
     * @param tags list of tags
     */
    public Item(String name, int id, LinkedList<String> tags){
        this.name = name;
        this.id = id;
        this.tags = tags;
        this.info = new LinkedList<>();
    }

    /**
     * This constructor accepts all fields that the item stores
     * @param name the item name
     * @param id the byte value
     * @param tags the list of tags
     * @param info the description as a list of strings
     */
    public Item(String name, int id, LinkedList<String> tags, LinkedList<String>
info){
```

```
        this.name = name;
        this.id = id;
        this.tags = tags;
        this.info = info;
    }

    /**
     * This function is used to rename an item if need be.
     * @param name the new name
     */
    public void rename(String name){
        this.name = name;
    }

    /**
     * This function clears the tags that this item has
     */
    public void clearTags(){
        this.tags = new LinkedList<>();
    }

    /**
     * This function adds a new tag to this item
     * @param tag the tag to add
     */
    public void addTag(String tag){
        this.tags.addLast(tag);
    }

    /**
     * Checks to see if this item has the given tag
     * @param tag the tag to check for
     * @return
     */
    public boolean hasTag(String tag){
        return this.tags.contains(tag);
    }

    /**
     * This function clears the items description
     */
    public void clearInfo(){
        this.info = new LinkedList<>();
    }

    /**
     * This function adds a new line to the description
     * @param info a string to display
     */
    public void addInfo(String info){
        this.info.addLast(info);
    }
}
```

Version 2: Tag.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */
package vendingmachinegui3;

import java.util.LinkedList;

/**
 * This case is a revised version of the earlier Tag.java
 * This holds cases as well as items
 * @author William Jones
 */
public class Tag {

    String tag;
    LinkedList<Case> cases;
    LinkedList<Item> items;

    /**
     * The constructor for this class
     * @param tag the tag to create
     */
    public Tag(String tag){
        this.tag = tag;
        this.cases = new LinkedList<Case>();
        this.items = new LinkedList<Item>();
    }

    /**
     * Adds an item to this tag
     * @param item
     */
    public void add(Item item){
        this.items.addLast(item);
    }

    /**
     * Adds a case to this tag
     * @param aCase
     */
    public void add(Case aCase){
        this.cases.addLast(aCase);
    }

    /**
     * produces a list of item names from the items in this case
     * @return list of names
     */
    public LinkedList<String> listItems(){
        LinkedList<String> items = new LinkedList<>();
        for(int i = 0; i < this.items.size(); i++){
            items.add(this.items.get(i).name);
```

```
        }
        return items;
    }
}
```

Version 2: TextFile.java

```
/**
 * This code is part of the Ambulance Interior Storage Optimization IPQ
 *
 */
import java.io.BufferedReader;
import java.io.FileReader;
import java.util.LinkedList;


/**
 * This class is used to read in a text file and store the lines as a linked list
 * @author William Jones
 *
 */
public class TextFile {

	LinkedList<String> content;


	/**
	 * The constructor for this class, it accepts a string with the path to the text file.
	 * @param fileName
	 */
	TextFile(String fileName){
		content = new LinkedList<String>();// initialize linkedList
		try{//try to read the file
			BufferedReader in = new BufferedReader(new FileReader(fileName));
```

```
                String text;
                while (in.ready()) {
                        text = in.readLine();// get line
                        content.addLast(text);// add line to the end of the list
                        System.out.println(text);// print the line to the console
                }

                in.close();// close the file reader
            }
            catch(Exception e){

            }
        }

}
```

Version 2: VendingMachineGui3.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */
package vendingmachinegui3;



import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.FileHandler;
import java.util.logging.Logger;
import java.util.logging.SimpleFormatter;


/**
 * This is the main class of the improved code
 * @author William Jones
 */
public class VendingMachineGui3 {
   /**
    * This function is the main function of the program
    * @param args the command line arguments
    */
   public static void main(String[] args) {
```

```
    /*
     * Set the path to the program files
     * This is the default path to the directory where config file resides
     */
    String path = "C:\\Users\\Will\\Desktop\\Java\\Workspace\\Files\\VM";

    /*
     * Set up Logger
     */
    Date date = new Date();
    SimpleDateFormat ft = new SimpleDateFormat("MM-dd-yyyy_hh-mm-ss");
final Logger logger = Logger.getLogger("Vending Machine");
FileHandler fh;
System.out.println(ft.format(date));

    try {
            // Create the logger
   // This block configure the logger with handler and formatter
   fh = new FileHandler(path + "\\Log\\Log_" + ft.format(date) + ".log");
   logger.addHandler(fh);
   SimpleFormatter formatter = new SimpleFormatter();
   fh.setFormatter(formatter);

   // the following statement is used to log any messages
   logger.info("Start of Log");
```

```
    } catch (SecurityException e) {
       e.printStackTrace();
    } catch (IOException e) {
       e.printStackTrace();
    }


        System.out.println("Start: File Test");
        logger.info("Read Config File");
        TextFile config = new TextFile(path + "\\Config.txt");

        /*
         * Set up data
         */
        final Index index = new Index(path, config);

        // TODO setup com ports

    /*
     * Setup the look of the gui
     */
    try {
       for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
          if ("Nimbus".equals(info.getName())) {
             javax.swing.UIManager.setLookAndFeel(info.getClassName());
```

```
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
    }
    //</editor-fold>

    /* Create and displays the gui */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new VMGUI(index).setVisible(true);
        }
    });
  }

}
```

Version 2: VMGUI.java

```
/**
 * This code is used for the Ambulance Interior Storage Optimization IPQ.
 *
 */
package vendingmachinegui3;

import java.util.LinkedList;

import javax.swing.DefaultListModel;
import javax.swing.JList;
import javax.swing.ListSelectionModel;

/**
 * This class is the gui for this version of the code
 * @author William Jones
 */
public class VMGUI extends javax.swing.JFrame {

        Index index;
        Tag selectedTag;
   /**
    * Creates the new gui
    * @param index the index of data to pass to the gui
    */
   public VMGUI(Index index) {
        this.index = index;//save the index
        // initialize the gui components
      initComponents(this.index.tagNames.toArray(new String[0]));
```

```
}

/**
 * This is the initialization function for the gui
 * @param tags the list of tags to display
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
 void initComponents(String[] tags) {
        final String[] tagList = tags;
        // initialize components
    jScrollPane1 = new javax.swing.JScrollPane();
    jList1 = new javax.swing.JList();
    jScrollPane2 = new javax.swing.JScrollPane();
    //jList2 = new javax.swing.JList();
    jScrollPane3 = new javax.swing.JScrollPane();
    //jList3 = new javax.swing.JList();
    jTextField1 = new javax.swing.JTextField();
    jScrollPane4 = new javax.swing.JScrollPane();
    jTextArea1 = new javax.swing.JTextArea();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jTextField2 = new javax.swing.JTextField();
    jScrollPane5 = new javax.swing.JScrollPane();
    jTextArea2 = new javax.swing.JTextArea();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    // set the program to close when the window is closed
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
// set the list of tags
jList1.setModel(new javax.swing.AbstractListModel() {
   String[] strings = tagList;
   public int getSize() { return strings.length; }
   public Object getElementAt(int i) { return strings[i]; }
});
jList1.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
jList1.addMouseListener(new java.awt.event.MouseAdapter() {
     //listen for mouse click
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jList1MouseClicked(evt);
   }
});
jScrollPane1.setViewportView(jList1);// set the list in a scroll pane
//  initialize the second list as empty
jList2.setModel(new javax.swing.AbstractListModel() {
   String[] strings = { };
   public int getSize() { return strings.length; }
   public Object getElementAt(int i) { return strings[i]; }
});
jList2.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
jList2.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jList2MouseClicked(evt);
   }
});
jScrollPane2.setViewportView(jList2);
// initialize the third list as empty
jList3.setModel(new javax.swing.AbstractListModel() {
```

```
    String[] strings = { };
    public int getSize() { return strings.length; }
    public Object getElementAt(int i) { return strings[i]; }
});
jList3.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
jList3.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jList3MouseClicked(evt);
    }
});
jScrollPane3.setViewportView(jList3);
// set text field as blank
jTextField1.setText("");

jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane4.setViewportView(jTextArea1);

jLabel1.setText("Case");

jLabel2.setText("Selected Item");

jTextField2.setText("");

jTextArea2.setColumns(20);
jTextArea2.setRows(5);
jScrollPane5.setViewportView(jTextArea2);
//create button for requesting a case
jButton1.setText("Request Case");//left button label
```

```
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });
    // create button for requesting a single item
    jButton2.setText("Request Med");//right button label
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });
    // gui layout
    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
Short.MAX_VALUE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jTextField1)
```

```
            .addComponent(jScrollPane4, javax.swing.GroupLayout.DEFAULT_SIZE, 300,
Short.MAX_VALUE)

            .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

            .addComponent(jTextField2)

            .addComponent(jScrollPane5)

            .addGroup(layout.createSequentialGroup()

              .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

              .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))

          .addContainerGap())

    );

    layout.setVerticalGroup(

      layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

      .addGroup(layout.createSequentialGroup()

        .addContainerGap()

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

          .addComponent(jScrollPane1)

          .addGroup(layout.createSequentialGroup()

            .addComponent(jLabel1)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

            .addComponent(jScrollPane4, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
Short.MAX_VALUE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
```

```
                .addComponent(jLabel2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jScrollPane5, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
Short.MAX_VALUE)
                .addGap(132, 132, 132)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(jButton1)
                  .addComponent(jButton2)))
              .addComponent(jScrollPane2)
              .addComponent(jScrollPane3))
            .addContainerGap())
      );

      pack();
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO copy functions to dispense a single item
    }

    /**
     * Function to update the gui when a tag is selected
     * sets the second list to the cases of the new tag and
     * sets the third list as empty
     * @param evt
     */
```

```
private void jList1MouseClicked(java.awt.event.MouseEvent evt) {

        String selection1 = (String)jList1.getSelectedValue();
        // find tag object
        for(int i = 0; i < this.index.tags.size(); i++){
                if(this.index.tags.get(i).tag == selection1){
                        this.selectedTag = this.index.tags.get(i);
                        listModel2.removeAllElements();
                        for(int j = 0; j < this.index.tags.get(i).cases.size(); j++){
                                listModel2.addElement(this.index.tags.get(i).cases.get(j).name);;

                        }

                }
        }


   // TODO code for updating the gui
}

private void jList2MouseClicked(java.awt.event.MouseEvent evt) {
   // TODO code for when a case is selected
}

private void jList3MouseClicked(java.awt.event.MouseEvent evt) {
   // TODO code for when an item is selected
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
        // TODO add function for requesting a case
    }


    /**
     * Unused main added by the netbeans editor
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(VMGUI.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                //new VMGUI(new Index()).setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JList jList1;
    DefaultListModel listModel2 = new DefaultListModel();
    private javax.swing.JList jList2 = new JList(listModel2);
    //private javax.swing.JList jList2;
    DefaultListModel listModel3 = new DefaultListModel();
    private javax.swing.JList jList3 = new JList(listModel3);
    //private javax.swing.JList jList3;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JScrollPane jScrollPane3;
```

```
    private javax.swing.JScrollPane jScrollPane4;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextArea jTextArea2;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    // End of variables declaration
}
```