STATE OF CALIFORNIA DEPARTMENT OF TRANSPORTATION

TECHNICAL REPORT DOCUMENTATION PAGE

TR0003 (REV. 10/98)

1. REPORT NUMBER CA-2012-1103	2. GOVERNMENT ASSOCIATION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE AND SUBTITLE Continued Research, Evaluation, and Implementation of Methods to Reduce Roadside Trash		5. REPORT DATE June 30, 2012 6. PERFORMING ORGANIZATION CODE AHMCT
7. AUTHOR(S) Matthew H. Jones, George W. Burkett Jr., Wilderich A. White, Steven A. Velinsky		8. PERFORMING ORGANIZATION REPORT NO. UCD-ARR-12-06-30-07
9. PERFORMING ORGANIZATION NAME AND ADDRESS AHMCT Research Center UCD Dept of Mechanical & Aerospace Engineering One Shields Avenue Davis, California 95616-5294		10. WORK UNIT NUMBER 11. CONTRACT OR GRANT NUMBER IA 65A0275, Task ID 1103
12. SPONSORING AGENCY AND ADDRESS California Department of Transportation Division of Research and Innovation 1227 O Street Sacramento, CA 94273-0001		13. TYPE OF REPORT AND PERIOD COVERED Final Report July 2008 – June 2012 14. SPONSORING AGENCY CODE CALTRANS
15. SUPPLEMENTAL NOTES		

16. ABSTRACT

Because roadside litter has become more problematic in California over the years, AHMCT has been contracted by Caltrans to research more effective and economical methods to reduce and remove litter from the roadsides. This has led to the development of the Debris Removal Attachment (DRA). The DRA will mainly be used to remove litter bags full of collected trash without requiring the operator to exit the vehicle, thus increasing worker safety, reducing the cost of equipment associated with a dedicated vehicle, and increasing the effectiveness of litter removal. This report documents the development of the control system, including all electronics, actuators, and control software. It also includes dynamic models for actuator selection, documentation of the assembly and testing, and a comparison between the design goals and measured performance of the prototype DRA.

17. KEY WORDS Road Maintenance; Litter; Debris; Automated	18. DISTRIBUTION STATEMENT No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.	
19. SECURITY CLASSIFICATION (of this report) Unclassified	20. NUMBER OF PAGES 180	21. PRICE

Reproduction of completed page authorized

Advanced Highway Maintenance and Construction Technology Research Center

Department of Mechanical and Aerospace Engineering
University of California at Davis

Continued Research, Evaluation, and Implementation of Methods to Reduce Roadside Trash

Matthew H. Jones, George W. Burkett Jr., Wilderich A. White &
Professor Steven A. Velinsky: Principal Investigator

Report Number: CA-2012-1103
AHMCT Research Report: UCD-ARR-12-06-30-07
Final Report of Contract: IA65A0275, Task ID 1103

June 30, 2012

California Department of Transportation

Division of Research and Innovation

ABSTRACT

Because roadside litter has become more problematic in California over the years, AHMCT has been contracted by Caltrans to research more effective and economical methods to reduce and remove litter from the roadsides. This has led to the development of the Debris Removal Attachment (DRA). The DRA will mainly be used to remove litter bags full of collected trash without requiring the operator to exit the vehicle, thus increasing worker safety, reducing the cost of equipment associated with a dedicated vehicle, and increasing the effectiveness of litter removal. This report documents the development of the control system, including all electronics, actuators, and control software. It also includes dynamic models for actuator selection, documentation of the assembly and testing, and a comparison between the design goals and measured performance of the prototype DRA.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DISCLAIMER

LIST OF ACRONYMS AND ABBREVIATIONS

Acronym	Definition
AHMCT	Advanced Highway Maintenance and Construction Technology
CALTRANS	California Department of Transportation
CLAP	Caltrans Litter Abatement Plan
DRA	Debris Removal Attachment
DRV	Debris Removal Vehicle

CHAPTER 1: INTRODUCTION

1.1 Introduction and Chapter Layout

This report documents the development of a Debris Removal Attachment (DRA) by the Advanced Highway Maintenance and Construction Technology (AHMCT) Research Center to increase the effectiveness and safety of roadside litter collection operations. This report documents the development of the electronics, the assembly and integration, and the testing of the prototype DRA. This chapter serves as an introduction to litter and debris litter collection operation and the DRA application, and begins with a discussion of the national and California state roadside litter problem. The need for automated litter and debris removal is established and several current methods to remove it are discussed, including both manual labor and other automated removal machinery. The need for a Debris Removal Attachment and its function are presented. Lastly, the chapter concludes with an overview of the remaining chapters in the report. Note that in this report the terms litter and debris are often used interchangeably.

1.2 The Need for Litter Removal

According to a 1993 report by Andres in The Synthesis of Highway Practice, 55.9% of the states considered interstate litter a “major” problem, while 35.3% considered it an “intermediate” problem [1]. The report stated that in 1993, the United States spent over $120 million to remove thousands of tons of litter from highways. Of the money spent, California was the top spender with a total of $28 million ($39 million in 2006 dollars). Thirteen years later, in the year 2006, the money spent by California for litter removal is reported as $55 million [2]. This is due to an increase both in the amount of litter being deposited and the cost to remove it. This demonstrates the importance of developing effective and economical litter removal procedures.

Litter deposit in California has become such a problem that in 2006, the Caltrans Litter Abatement Plan (CLAP) was created to address the growing concern of litter and illegal dumping in California [2]. CLAP states that litter deposit on highway roadsides has a negative effect on environmental, social, and economic issues. CLAP is a comprehensive approach to address litter issues that includes a balance of different approaches to litter abatement. According to CLAP, all effective measures to reduce litter have three common elements: (1) public education and awareness (preventative), (2) litter control and removal (reactive), and (3) litter law enforcement (corrective).

In the Caltrans Litter Abatement Plan, litter is defined as

- All trash, cigarette butts, refuse, junk, garbage, or scrap,
- Any articles or material deposited within the right of way, intentionally or unintentionally, or
- Any article or material abandoned by the owner or the person in possession thereof, not including dust, smoke, or other like products emitted or produced during the normal operations of any mining, extractive, primary or manufacturing industry.

The term "litter" is most appropriately used for spontaneous or unintentional acts that involve objects of smaller size and quantity, while "illegal dumping" is most appropriately used to describe premeditated acts including objects of larger size and quantity. Only clear water and feathers from live birds are allowed to exit a vehicle and be left on the roadside [2].

Litter most commonly removed from California's highways includes [3]:

- Paper, bottles and cans
- Tires
- Refrigerators and stoves
- Mattresses
- Rugs
- Ladders

About half of all containers found on our highways contained alcoholic beverages. According to CLAP, the primary sources of litter are

- Drivers
- Illegal dumping
- Trucks with improperly covered loads
- Pedestrians
- Construction sites
- Demolition sites
- Household garbage cans
- Commercial dumpsters
- Industrial sites
- Loading docks

The desired results of CLAP are to reduce litter, improve the appearance of the environment, be responsive to local and regional needs, and to increase the public's litter awareness and anti-litter education.

1.3 Current Manual Litter and Debris Removal Methods

The typical process to remove litter from the roadside has two distinct phases: (1) collection of the roadside litter into litter bags and (2) removal of the litter bags from the roadside onto a transport vehicle [4]. Usually the filled bags are collected by a couple of Caltrans workers from the local maintenance yard using a ¾ - 1 ton truck. They drive up to the bags, exit the vehicle, throw the bags into the truck bed, hop back in and move forward to the next pile. Once back at the yard, the debris is transferred to a bin.

Much litter is collected and bagged by Adopt a Highway volunteers and probationers instead of Caltrans workers. This is a significant cost saving to the state. A crew of a dozen people will be supervised over the course of the day to clear a few miles of road side and the bags and associated debris will be set along the shoulder as shown in Figure 1.1.

The local Caltrans maintenance yard has the responsibility to collect the bags and dispose of them properly. Often the bags are left along the road for a few days as a reminder to the traveling public of the effort required to collect the litter. This also gives the yard flexibility in scheduling their collection effort.

Ideally litter collection occurs on a regular basis before an unsightly buildup. In recent years Caltrans has implemented extended agreements with local agencies to use probationers on a more regular basis. In the ideal case of regular collection, the end result may be as few as a single bag every tenth of a mile. Periodically a larger piece of debris such as a tire carcass or piece of cardboard will be placed along the bag. Regular collection greatly improves the appearance of roadsides but also requires more frequent bag collection by Caltrans crews.

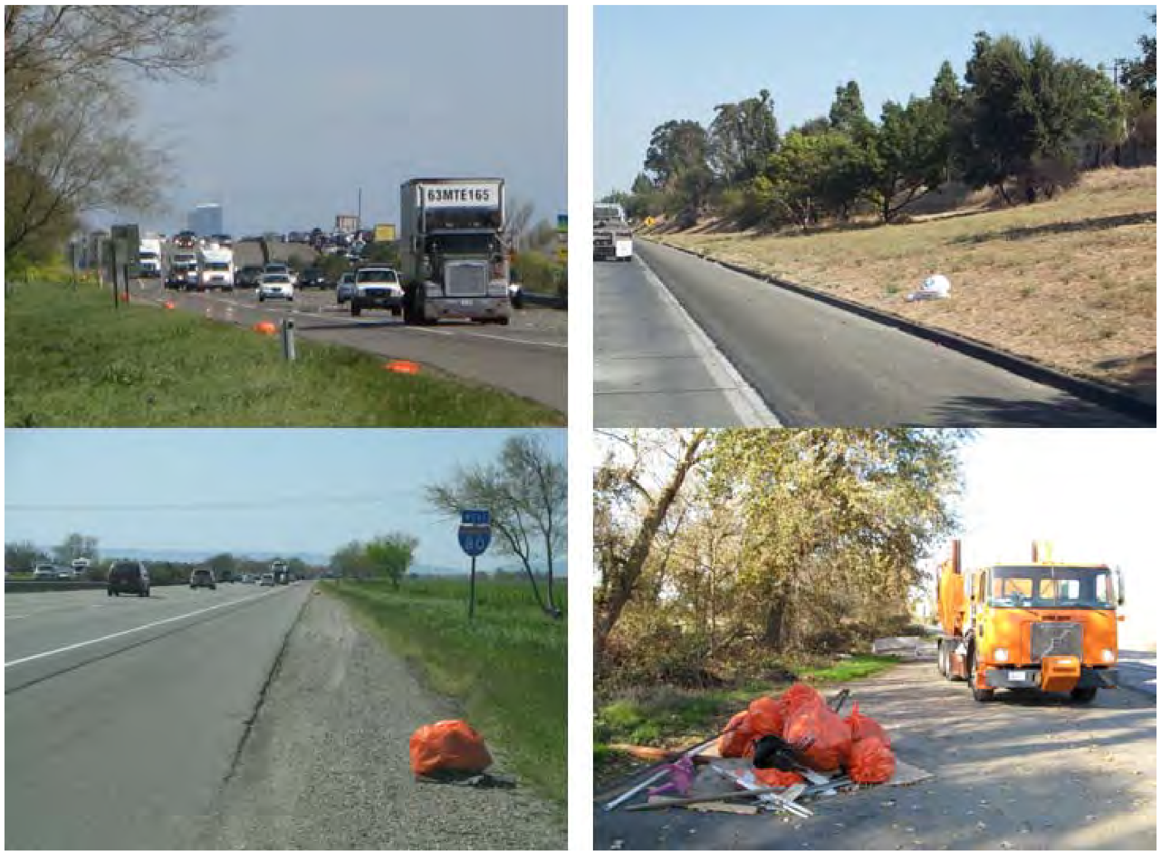


Figure 1.1: Bagged litter and loose debris collected for retrieval by Caltrans crews

Dumping of trash by the public occurs frequently along some roads. The mattress on the lightly traveled road shown in Figure 1.2 may have been improperly tied down but was more likely purposely dumped. In this environment, the bagging of loose litter is less common but Caltrans must still regularly collect debris. This type of material is usually collected manually on an as needed basis and can consist of heavy pieces.

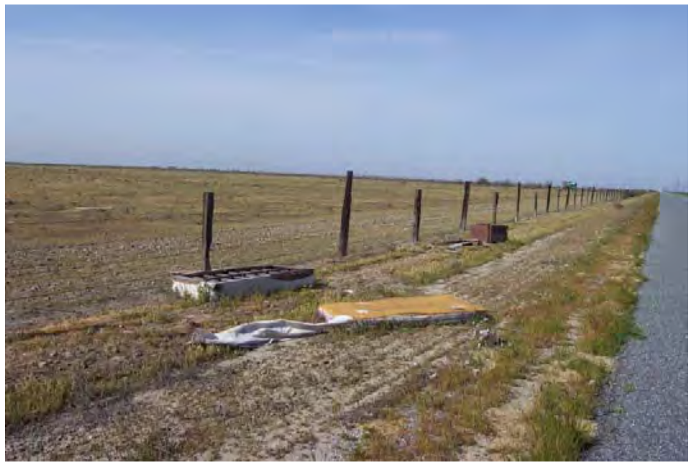


Figure 1.2: Mattress and bed that was dumped

1.3.1 Equipment Used in Debris removal

Difficult to access or hazardous areas in urban regions or often collected less frequently and the resulting debris piles become large and more challenging to remove. In these situations, Caltrans will use heavy equipment such as a loader and dump truck to remove the debris. In Figure 1.3, a rear loading refuse truck is operating along a median which requires lane closures. This type of clean up will involve a large crew and may include additional support equipment to remove mowed weeds and vegetation trimmings. The litter collected will be bagged and thrown into the back to the truck.

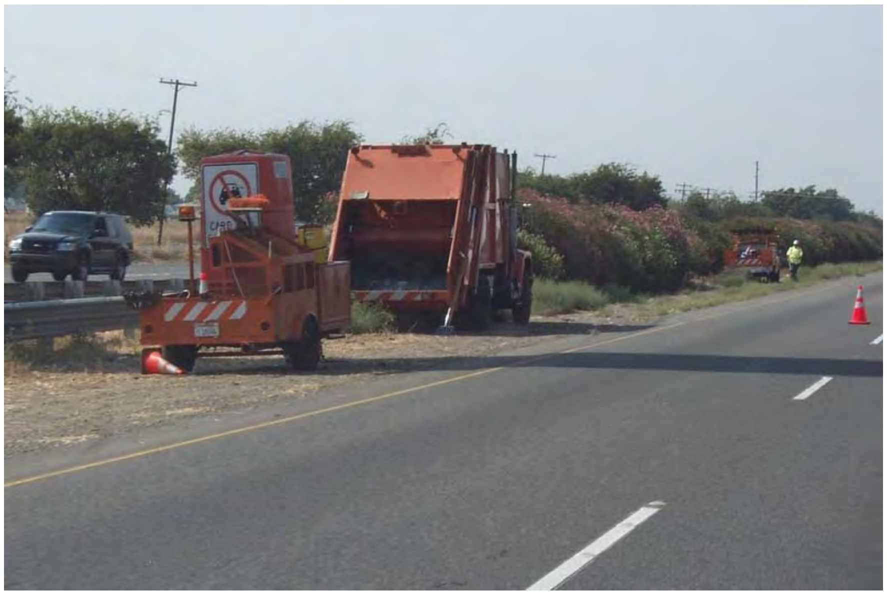


Figure 1.3: Caltrans end loading refuse truck

Litter collection in the more hazardous locations is directed and/or performed by Caltrans crews using the necessary shadow vehicles and lane closures. It is usually done in conjunction with other more critical maintenance work.

A unique piece of equipment in the Caltrans fleet that is used for debris and litter collection is the dump body truck with integrated loader shown in Figure 1.4. The one cubic yard bucket has a clam shell to clamp down on what is scooped up. It will raise items from in front of the truck over the cab and dump them into the dump body. This vehicle is typically used to collect tire carcasses, which are very common debris (Figure 1.5) along the high speed freeways and especially in the hotter desert areas.

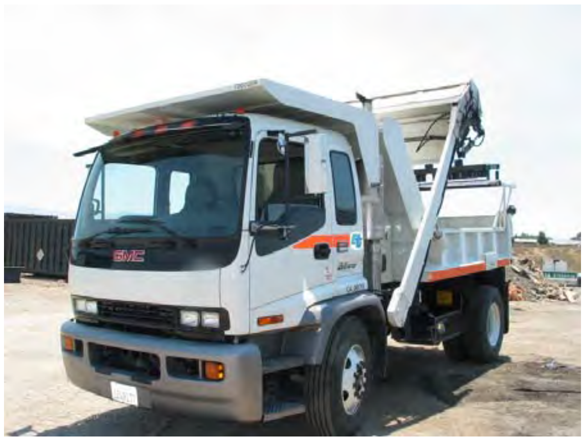


Figure 1.4: Caltrans dump body with loader

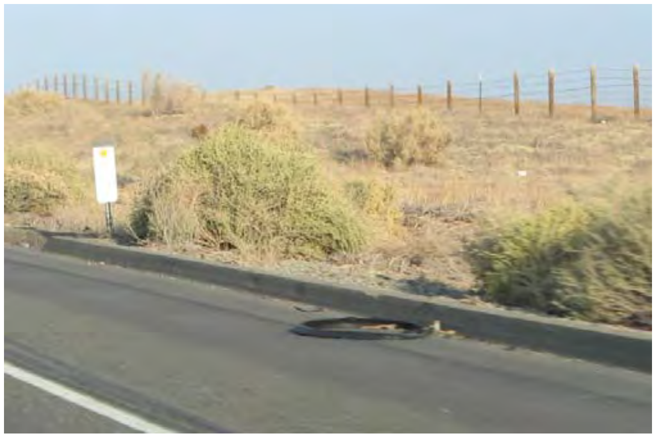


Figure 1.5: Piece of tire carcass common on high speed roads in the desert

Although not common in Caltrans fleet, vehicles with grapplers can be used for debris collection. A typical truck mounted grappler is shown in Figure 1.6 and a less common trailer mounted unit is shown in Figure 1.7.

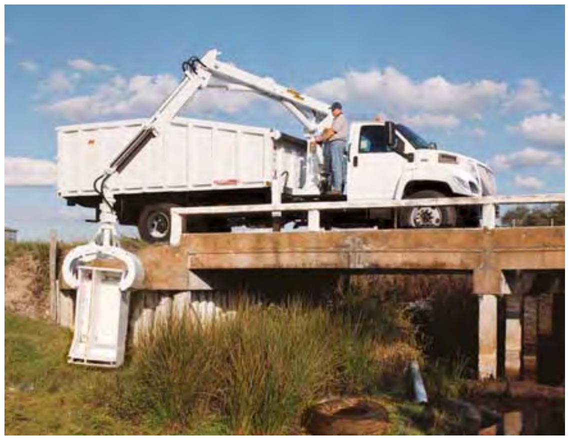


Figure 1.6: Large grappler system, (image from Wikipedia)

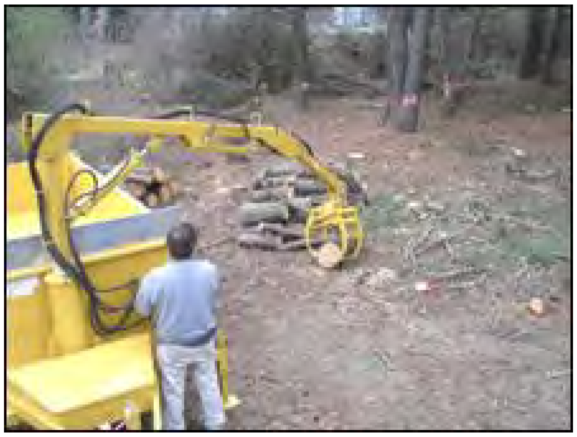


Figure 1.7: Small grappler from Centreville Manufacturing in Maryland

There has been long term interest in Caltrans to incorporate this type of articulated arm onto a truck that can be used for roadway litter bag and debris collection. Workers cannot ride on the back of a vehicle in traffic. The design challenge is to place the operator controlling the arm within the cab.

1.4 Automated Debris Removal

A critical reason for automating Caltrans operations with advanced machinery is to increase the safety of Caltrans workers. Since the year 1924, 172 Caltrans workers have died on the job, most of them struck by inattentive or reckless drivers passing through work zones [5]. To improve safety, it is important to remove the worker from the roadside by developing machinery that can be controlled from the safety of inside the truck [4]. Automated litter removal procedures can also help increase the effectiveness of litter removal, which can help reduce the cost [2, 4].

A recent death that highlights the hazards to workers was that of Donald Lichliter on July 23, 2009 [6]. Lichliter was on the roadside of Highway 99 near the Kettleman Lane exit in Lodi, when a commercial glass truck struck and killed him. Lichliter was following the routine procedure of spreading fertilizer on eucalyptus trees when the accident occurred. The development and effective deployment of automated roadside equipment, like the DRA, can help minimize stories like Donald Lichliter's.

CLAP also calls for mechanical innovations that enable litter to be more effectively and economically collected from the roadside [2]. These include systems that are currently available or that can be developed and implemented. CLAP states that the deployment of mechanical devices will maximize Caltrans investments, "thereby enabling the department to contribute towards a cleaner environment in California."

Automating the debris removal process can simultaneously increase the effectiveness, economics, and safety of roadside debris removal operations. Removing the workers from the roadside with the use of automated machinery is a key step in increasing worker safety.

A dedicated debris removal vehicle (DRV) was previously developed by Caltrans and AHMCT for roadside litter bag collection [4]. The DRV was based on a conventional garbage compacting truck and has a hydraulic clamshell type manipulator (see Figure 1.8). The early DRV prototype had the ability to pick up debris from either side of its 7.6 cubic meter (10 cubic yard) compactor, and deposit the debris into the compactor. AHMCT implemented a control system that optimized the use of the arm and automated various motions. The clamshell manipulator had a basic level of automation consisting of several preset arm deployment positions and an automated dump sequence. The DRV required the operator to maneuver the arm to the exact location of the debris, close the clamshell, and initiate the dump sequence. All of the human control was done from inside the cab of the truck, reducing the need for workers on the roadside.

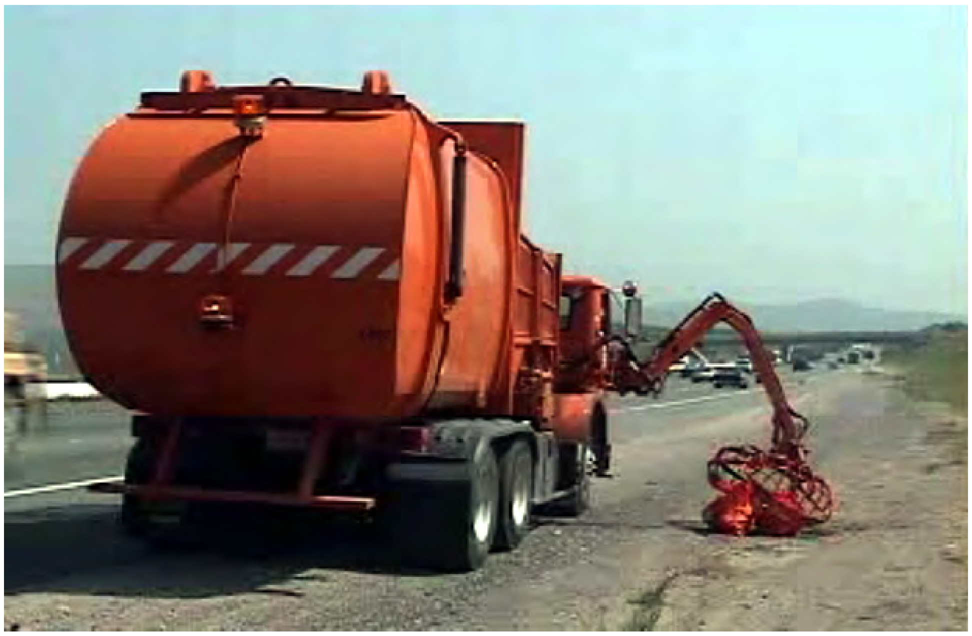


Figure 1.8: Dedicated Debris Removal Vehicle with Hydraulic Arm System [4]

Caltrans revised the design to incorporate a commercially available arm used on residential trash collection trucks. Figure 1.9 shows this latest revision of the DRV system.

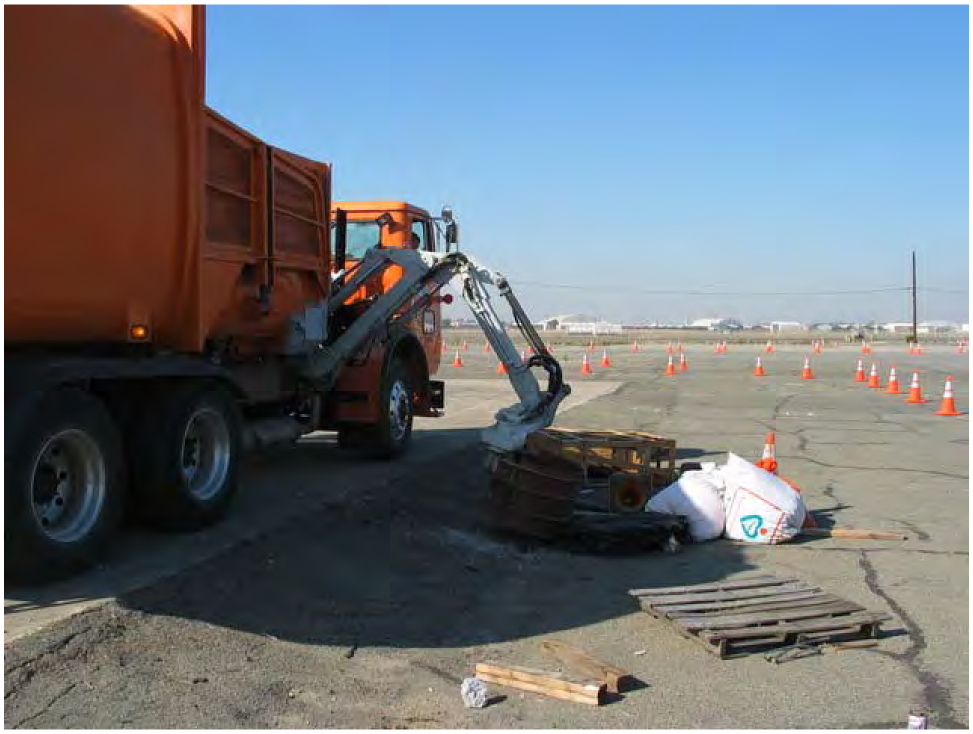


Figure 1.9: Latest revision of the Caltrans DRV is integrated with commercial arm

Field testing of the earlier DRV demonstrated that it was capable of picking up several tires or up to 8 bags of litter in one grab and dump sequence. That DRV was also able to pick up mufflers, lumber, and other large debris up to 45 kg (100 lbs) but a redesign of the arm was required to improve robustness. The latest version has a much heavier lift capability but the clam shell action is significantly limited by the arm geometry, which was optimized to grab a trash can.

Although less dexterous than the earlier prototypes, the function of the latest DRV was adequate for many of the litter bag and debris collection operations. Collection of bags is most often performed on the right side of the highways and litter collection crews could eventually be trained to place bags in reachable locations. Some refinements were required to improve reliability and acceptance by the operators was limited by these problems. In order to use the DRV cost effectively, a dedicated operator is required. Given the low numbers of these vehicles that would be needed initially, no vendors have been willing to invest in further development.

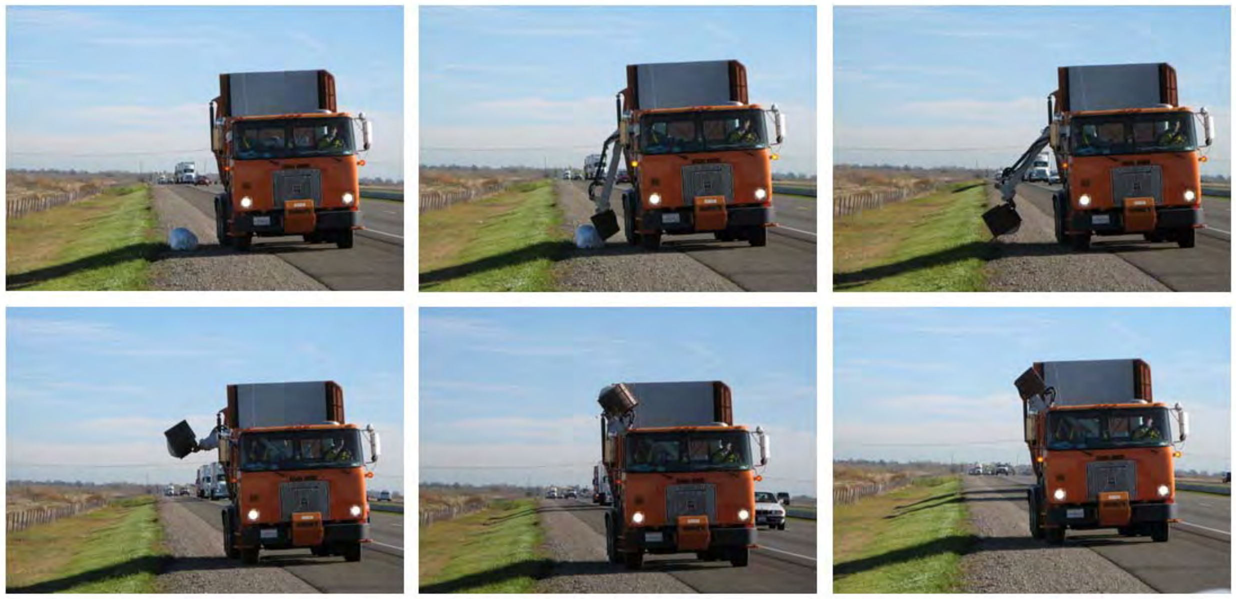


Figure 1.10: Caltrans DRV deployment in 2007

1.5 Advantage of the DRA

The disadvantage of the DRV is that it is a dedicated vehicle, which greatly increases its purchase, maintenance, and operation costs. Caltrans is severely restricted in the total number of vehicles in its fleet. AHMCT developed the DRA concept as a low cost alternative to the DRV concept, which will potentially allow multiple units to be purchased and deployed for the benefit of Caltrans workers.

1.6 Summary and Chapter Layouts

Roadside litter has a major issue in California over the last several years. The money spent by Caltrans to remove the litter doubled in just a 13 year period; from $28 million to $55 million. It has thus become increasingly important to develop methods to effectively and economically remove this litter. To increase the safety of roadside operations, it is essential to develop advanced automated machinery to replace workers on the roadside. The Debris Removal Attachment can enhance the safety and efficiency of roadside litter bag collection while overcoming most of the disadvantages of a dedicated Debris Removal Vehicle.

Chapter 2 describes the development of the DRA concept and the final design. Chapter 3 documents the requirements and selection of the DRA's electronics including the motor driver electronics, logic control, and sensors, and explains how they will be used together to control the DRA. Chapter 4 continues with an explanation of the DRA's cycle time requirements and other motor constraints, and explains in detail the dynamic models used in the motor selection procedure. Chapter 5 discusses the DRA's logic control needs and gives a high level overview of the logic control unit's programming. Chapter 6 describes the assembly, integration, and initial testing of the DRA. The report ends with conclusions and recommendations for the DRA in Chapter 7.

CHAPTER 2: Development of the DRA Concept and Prototype

2.1 Design Process

This chapter describes the design development from concept selection through final prototype design. The process of design development begins with a complete assessment of the debris collection problem and definitions of design requirements. Collection of debris in the roadway and on the roadside are considered. Roadside collection is selected as the most important function. As would be expected, the final design shares similar features with the DRV described in the previous chapter.

2.2 Classification of Debris

In the development of a debris removal system, it is important to understand two major aspects of the removal process. This project is not going to look at very small litter like small pieces of paper. This project is focused on larger debris. However, these larger items can be organized based on their geometric location relative to the travel way. Each class has very distinct issues that significantly affect the deployment of the system.

The first class of debris to discuss is composed of large objects that are in the travel way. These items are typically large objects which have fallen off of vehicles in the roadways. Smaller items such as papers typically migrate to the shoulders. The larger objects pose an immediate risk to the public and must be removed promptly. These objects are typically removed from the travel way by first setting up a temporary emergency closure and then manually removing it. There are a few key aspects of the kind of debris which should be noted. First, the kind of debris is very random in terms of physical size. Therefore any system to deal with this issue must maximize it’s flexibility in terms of what it can pick up. Second, since this poses an immediate risk to the public the system must be able to be quickly deployed. This kind of debris is typically isolated and therefore the system does not require a huge amount of capacity.

The next class of debris is located on the shoulders of the travel way. Bags of litter are the most common type of shoulder debris. Cleanup crews collect small litter (such as paper and cans) in bags, which are left along the travel way’s shoulders. These bags typically weigh on the order of 20lb, with an occasional larger item. The way this is currently dealt with is that after the cleanup crew has bagged the smaller litter, a Caltrans crew goes out and picks up the bags and other debris items. Typically this is done by pulling up to the debris and manually picking up the items. Once everything is picked up, the truck moves to the next debris pile. There is a huge exposure risk to the worker by getting in and out of the vehicle numerous times. There are a few key aspects to note about this kind of debris removal process. One is that since cleanup crews bag the litter, the pickup can be pre-planned. Second, there is better consistency in the size and shape of this kind of litter as most of it is typically litter bags. It should also be noted here that any system would need to have a large capacity to ensure that all the debris can be easily collected.

2.3 Design Requirements

AHMCT developed a Debris Removal Attachment that will help automate the second phase of litter collection: the removal of litter bags and other large single item debris [7]. The key design goals of the DRA are shown below:

1. Increase worker productivity and safety
    a) Keep workers in the vehicle
    b) Make it simple to operate
2. Develop a Low cost system
    a) Simple and minimal machine design
    b) Easy and inexpensive to purchase and maintain
3. Integrate solution as attachment to current fleet
    a) No expensive dedicated vehicles
    b) Minimal impact on installed vehicle's present functions
    c) Power off of a standard vehicle 12V battery system

By designing a system that adheres to these goals, AHMCT believes that the benefits of an automated debris collection system can be realized without the disadvantages of the dedicated DRV.

Designing the system as an attachment to current vehicles by eliminating the need to purchase and maintain a dedicated vehicle greatly reduces the costs to implement the system [7]. Keeping the mechanics of the system simple also helps keep the purchase and maintenance costs down. The low cost of the DRA system will increase the number of units that can be purchased and deployed, thereby increasing productivity. Allowing the system to attach to current vehicles will add a new function to the vehicle without impacting any of the truck's other functions. The unit can also be removed, switched between sides, or swapped between vehicles if needed.

A 12V electrical drive system was selected for the DRA over a hydraulic system because an electrical system can easily integrate into a truck's power supply without the need for additional hydraulic pumps or separate power supplies. An electrical system allows for a lower cost proportional control and smoother operations.

2.4 General Concepts

After looking at some of the early design requirements various concepts were developed and presented to Caltrans. Early on, it was recognized that a single solution for the two fundamental classes of debris was not reasonable. The preliminary concepts presented here are separated out into two groups to address the two different classes of litter. The idea was to present potential solutions to both problems and get direction from Caltrans as to which debris class had the greater need and therefore the greater possibility of acceptance within the Caltrans maintenance procedures.

The first class that was looked at was travel way debris. Early on, it was felt that the best way to deal with this kind of litter was by having a front mounted system. This would allow a forward approach to the object, which would be compatible with the general flow of traffic. Since this kind of litter is very isolated, it stands to reason that a system which will simply deal with one item at a time would be sufficient. The course of action is to primarily focus on getting the debris out of the travel way. The simplest course of action would be to simply move the debris to a safer location on the shoulder to be dealt with manually. The cost of this system is a key concern as the number of required units is more critical with this type of operation. This is due to the fact that the system must respond quickly to the immediate public threat and therefore having a large number of distributed units is much more critical.

The first concept utilizing this method was to build an attachment that would interface with a standard small plow. The basic concept is presented below in Figure 2.1.

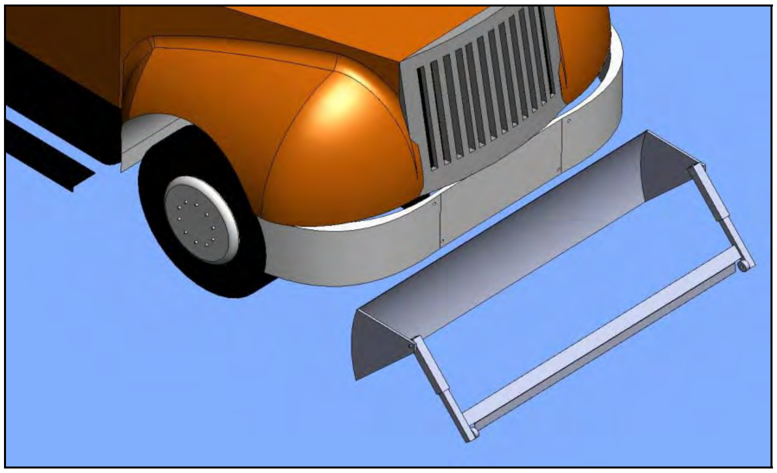


Figure 2.1: Modified plow attachment

The system would operate like a standard plow. Once the system was close to the target, the blade would be dropped to the roadway surface. While the item is being pushed forward, a bar would rotate in place to provide a positive force to push the debris against the blade. Then the item would be pushed off of the side of the road where it could be dealt with in a less hazardous location. The major benefit of this system is that it would build on platform which is already integrated with many Caltrans fleet vehicles. The drawback is that the blade will not be able to lift the item off of the roadway surface. This would cause damage to the roadway as many of the raised pavement markers would be damaged in the process. A large bulky blade on a vehicle on all times would hamper the mobility of the vehicle for other purposes. The key to this system would be to design the bar so that it can reach over and around a majority of items that it would encounter.

Another design concept was to make a system which would function similar to a dustpan. The basic concept is presented below in Figure 2.2.

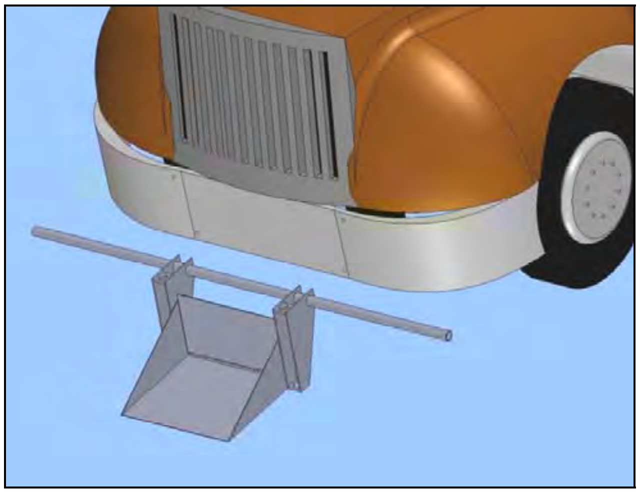


Figure 2.2: Front mounted scoop

The system would be designed to be folded up in a very compact manner when not deployed. This would minimize any adverse affects to the vehicle during normal operations. There would be an actuator to adjust the pitch of the system which could also aid in lifting the debris off the roadway. The general procedure would require the operator to drive close to the item and pull off to the shoulder prior to it. At this point, the system would be manually unfolded by the operator. Then the vehicle would drive to the area and lower the scoop to the roadway just prior to the debris. Once the item is “scooped” up, the system would be able to rotate slightly to lift the system off the ground. Since the system has the ability to lift the object there would be minimal chance for additional damage to the roadway surface. The notion of keeping the system fairly compact would impose limitation on the size of debris which could be picked it. Also, there is no positive force on the item, which would mean that awkward items have the potential of becoming loose again. This system relies on impact mechanics to scoop up the item. This seems contradictory to a lightweight system.

The other concept that was presented was to develop a foldable system that had a pair of jaws that would provide a positive force to capture the object. The basic concept is presented below in Figure 2.3.

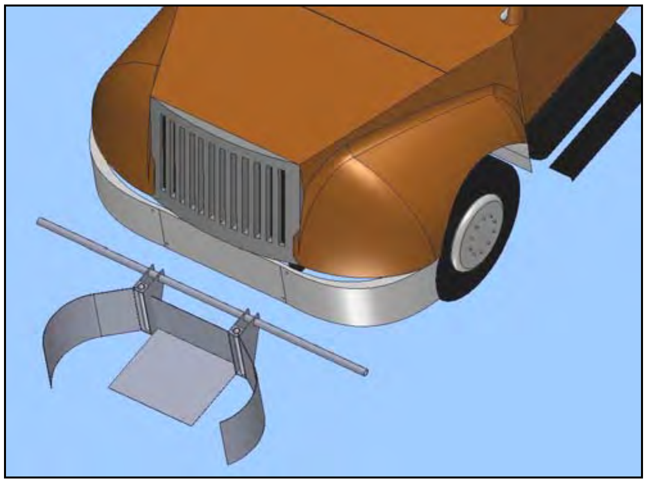


Figure 2.3: Front mounted jaw mechanism

The idea would be similar as the second concept, the primary difference is that any issues created due to the impact mechanics is reduced because the system can rely on the jaws to push items on the pan. The major drawback to this configuration is that the system would either be highly restrictive on what it could pickup in terms of physical size or it would be relatively large. The main reason behind this is that the greater number of required actuators would increase the overall size of the system.

The second group of conceptual designs focuses on debris located along the shoulder of the roadways. In this situation, a side mounted system is the optimal geometry. There may not be a clear approach from the rear of the objects as the shoulder width is not consistent. The shoulders may be soft and unable to support with weight of a heavy vehicle and a side mounted system would reduce how far the vehicle must divert from the travel way to pick up the debris. The attachment/un-attachment process should be kept as simple as possible to maintain the functionality of the host vehicle. Since the vehicle must be compatible with typical traffic, the system cannot significantly stick out of the vehicles frontal profile.

All of these concepts utilize the same general operation. The system has a transport mode, which allows it to integrate with general traffic with minimal effort. Once the vehicle is near the work zone area, it goes into a deployed mode. The system then moves into a position where is can capture the debris. Once the debris is captured, the system is elevated to a height which is located above the bed rails of the truck and the end effector is rotated until gravity takes control and drops the debris in the back of the vehicle. In all the concepts presented below there are two major parts to the system. The first part is the end effector which captures the debris. The second is the vertical/dump system which lifts the debris up and dumps it into the truck bed. These two systems can be decoupled and can be mixed and matched from the concepts presented below. The first concept consists of a drop box with a single track for the required vertical and rotational movement. This concept is illustrated below in Figure 2.4.

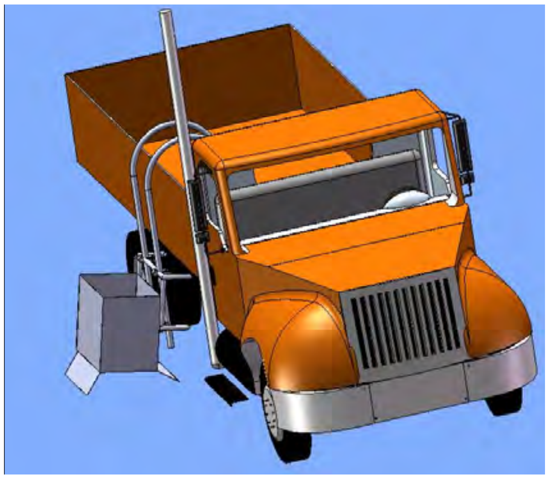


Figure 2.4: Horizontal jaws and a track which allows for vertical and rotational movement.

One of the benefits to this style of end effector is that the litter is picked up from the top. Often times when crews pickup litter they place bags right next to fixed objects. This makes having a system that captures from above more desirable. However, this box concept imposes a lot of restrictions on the object size, which could be a problem for more awkward items. The box type geometry increases the potential that the bags will become jammed within the system.

In an effort to simplify the number of required actuators in this design, the lifting and rotation required to dump the objects would be handled by a single motor. The drive system significantly increases the complexity of the system's bearing mechanics. Bearing systems which have both linear motion and rotational motion are fairly complex.

The concept presented below utilizes a scooping process to capture the debris and a linear guide/cam system to lift and rotate the end effector. The basic design is presented below in Figure 2.5

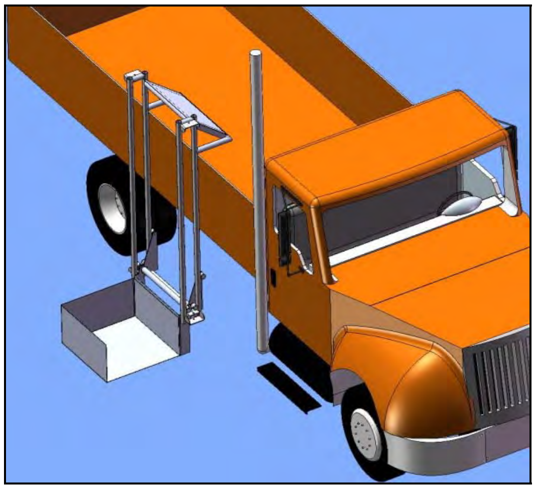


Figure 2.5: Scooping system with a vertical track and cam system.

The scooping system would require the system to be completely lowered prior to the item of interest. The forward movement of the vehicle would cause the bag to become captured in the scoop system. Then the system would be lifted up until the cam system engages and rotates the system. The fundamental benefit of the scooping system is that the end effector mechanics are very simple. This system would only require 1 actuator as the end effector does not require one. The operator of this system would only have to worry about lateral vehicle position. The longitudinal position is much less of a concern due to the end effector design. The bearing mechanics of the vertical/rotate system is much simpler in this case compared to the previously discussed system. The major drawback is that the impact mechanics would create a greater potential for ripping open the bags. The system must also be compatible with traffic, which would mean that the cam system would require almost 180 degrees of rotation. This is a significant range of motion for a basic cam type mechanism.

The last concept which was presented was to utilize a clam type system and have independent motors for the lifting and rotation axes. The concept is presented below in Figure 2.6.

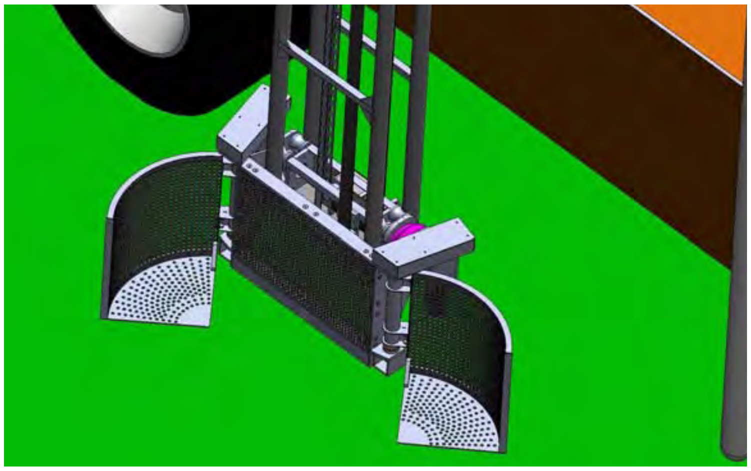


Figure 2.6: Clam end effector with mechanically independent vertical and rotational drives

One of the benefits to this system is that the actuators utilized to drive the system up and the drive the rotation are independent can be better optimization for a specific motion. The drawback to this system is that the control is much more complicated. First, there is no mechanical correlation between the vertical motion and the rotational motion that is inherent to the physical device. The control software must co-ordinate these two actuators.

These concepts were presented and discussed with Caltrans. The primary goal of these concepts was to initiate a dialog with Caltrans in order to better understand the immediate needs. This process helped to refine the projects goals and ensure there the system which is developed meets the needs of Caltrans.

2.5 Concept Selection

After presenting the various concepts, discussions were made in order to identify the future direction of the project. The first question which needed to be answered was what class of debris should be the focus of this project. In communication it was felt that the removal of shoulder debris had greater applicability. The benefit to this is that the size of the items is generally better understood as the bags typically weight about 20lbs and have a fairly consistent geometric shape. It was decided that the system should have the ability to lift up to 150lb to handle debris other then bags. The system cycle time is important to consider since it is directly related to system productivity. A reasonable cycle time of approximately 15 seconds seemed to be a reasonable target for this application.

Once the decision on what class of debris to focus on was made, discussion was made over the configuration. While looking at the various concepts in more detail, the last concept (Figure

2.6) became apparent as the most viable option. In terms of control, it is the most complicated. However, mechanically it is fairly straight forward. The design lends itself to better actuator optimization, which is the greatest factor in optimizing the system's cycle time. The jaw mechanism seemed to be the most reasonable as well. First, having a positive force on the debris logically seemed to be appropriate. Secondly, the jaw type configuration is more flexible in debris size then the other options. There is also the benefit that the gripping mechanism is similar to the system used on the dedicated DRV discussed above, which seems to work reasonably well in application. Once the general concept was defined, the detailed design was developed.

2.6 Proposed System: Debris Removal Attachment

2.6.1 DRA Concept Design

The concept system is shown in Figure 2.7[7]. The system consists of four main parts: a stationary frame, a shuttle, a dump assembly, and a gripping clamshell system. The back of the stationary frame attaches to the side of a truck bed and functions as a vertical track for the shuttle. The shuttle has rollers which ride between the tubes, the track of the stationary frame, and serves as a rotation base for the dump assembly. The dump assembly rotates 180^{o} about the long horizontal axis of the shuttle when it is at the top of the stationary frame, and dumps the payload over the back of the system into the truck bed. The clams are attached to the dump assembly and can be actuated independently to scoop under or positively grip a bag of litter. The block diagram in Figure 2.8 represents the high-level mechanical architecture of the DRA system. The shuttle, dump assembly, and clamshell system make up the dynamic assembly of the DRA.

DRA has a total of four electric motors: one for the vertical motion of the shuttle, one for rotational motion of the dump assembly, and two for each independent clam (left and right). The DRA design includes separate motors for the left and right clams to provide additional control not possible with a joint motor, and to eliminate the need for additional mechanisms to transfer joint motion between the two clams.

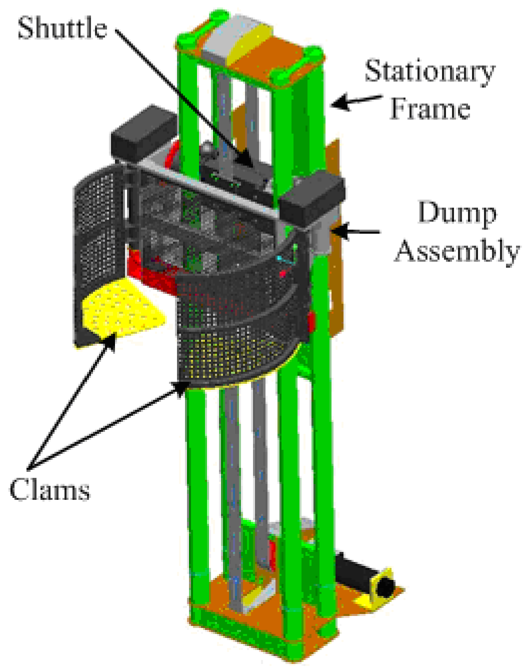


Figure 2.7: Debris Removal Attachment [7]

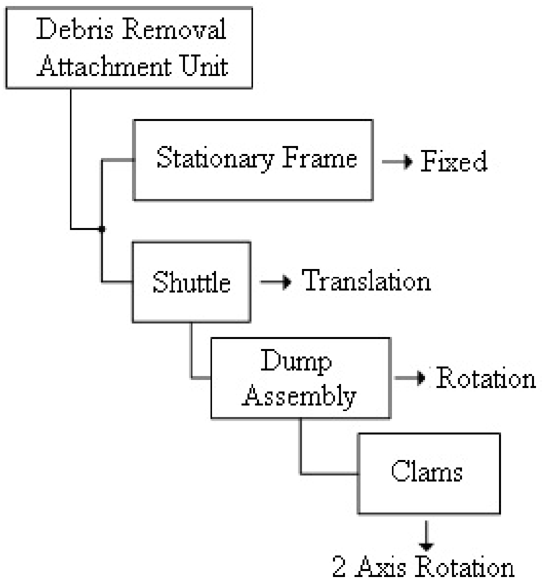


Figure 2.8: Block diagram of DRA system mechanical architecture [7]

The stationary frame serves four main functions: to provide a vertical track for the shuttle, to provide a means to drive the shuttle vertically, to serve as a load bearing element for the shuttle, and to serve as a mount to the side of the truck. The vertical track and load bearing element of the stationary frame consists of 4 vertical tubes. The shuttle has 8 u-groove wheels (4 on each side) which guide it between the 4 tubes and support both radial and axial wheel forces. The 4 wheels on each side are capable of supporting a significant amount of torque about the dump assembly axis. Three views of the tube and u-groove wheel mechanism are shown in Figure 2.9.

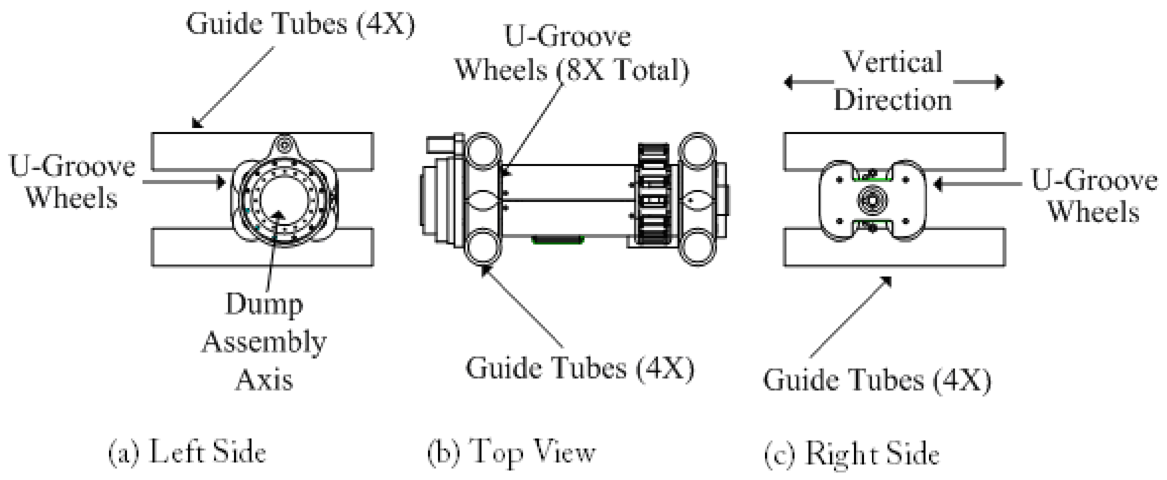


Figure 2.9: Stationary Frame Tube and Shuttle U-Groove Wheel Assembly [7]

The shuttle is driven vertically by a high-strength timing belt that runs continuously through the entire length of the vertical tubes, with the vertical motor located at the back of the base of the stationary frame. Several idle pulleys guide the belt, which is constrained to the shuttle by a clamp. The timing belt ensures zero slipping along the vertical path. The idle pulley closest to the motor drive pulley serves to increase the number of teeth engaged in the drive pulley. A cutout view of the vertical track and timing belt system is shown in Figure 2.10. This figure depicts the system lying out horizontally with what is normally the front of the unit facing up.

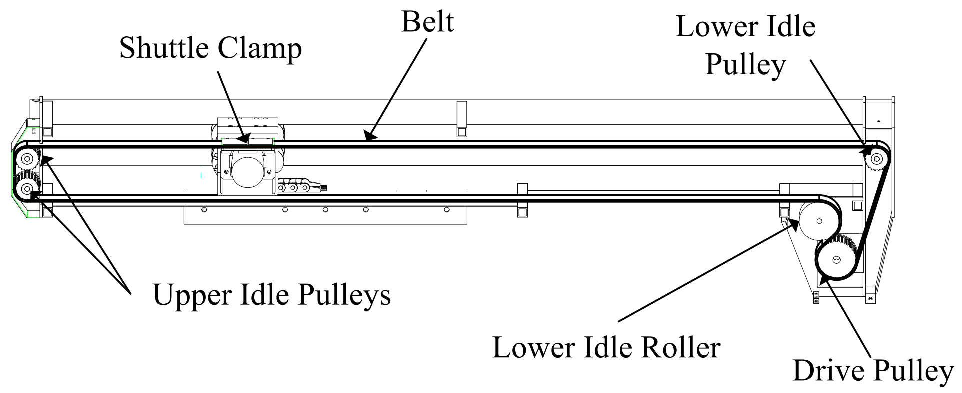


Figure 2.10: Vertical Track and Timing Belt Assembly (Front Facing Up) [7]

The electrical wires are passed from the back of the stationary frame to the moving shuttle by means of a cable carrier. A cable carrier is a linkage device that guides flexible wires connected to linear moving structures (for an example, see Figure 2.11).

The shuttle houses the rotation motor and serves as a rotation base for the dump frame, which rotates the payload over the DRA and into the truck bed. The torque of the rotation motor is transferred to the dump assembly through a harmonic gearbox, which also serves as the load bearing element on the left side of the shuttle. Electrical connections are transferred from the shuttle to the dump assembly through a slip ring, a mechanical device that allows electrical connections to pass from a stationary structure to a continuously rotatable structure (see Figure 2.12). For the DRA, the stationary structure is the shuttle and the rotating structure is the dump assembly. The slip ring ensures that no wires are required to twist, which could present fatigue issues. The shuttle assembly and its components are shown in Figure 2.13.

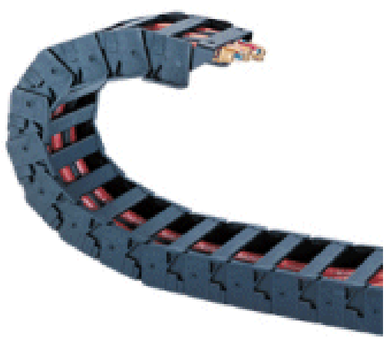


Figure 2.11:Example cable carrier [8]

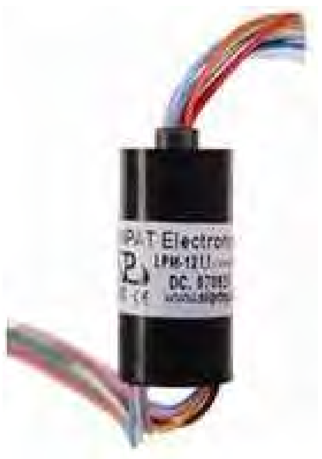


Figure 2.12: Slip ring [9]

The dump frame serves as a platform for the clamshell system. Since the left and right clams are independently actuated, they each have their own motor and drive mechanisms. The drive mechanism consists of the motor turning a geared pulley linked to the clam axis through a small timing belt. The entire dump assembly with the clamshell system is shown in Figure 2.14. The clam drive mechanism is shown in Figure 2.15 with the timing belt hidden.

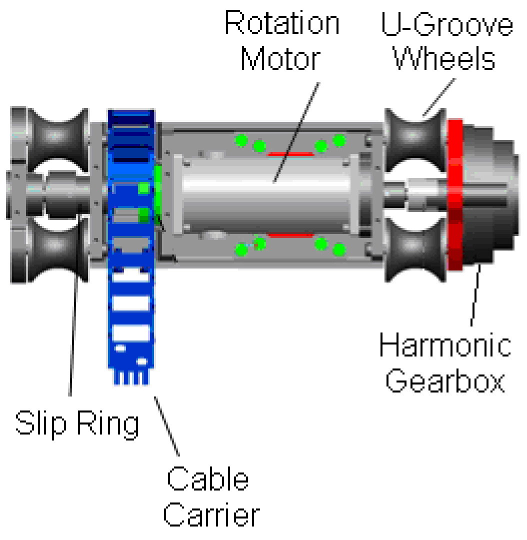


Figure 2.13: Shuttle Assembly and critical components [7]

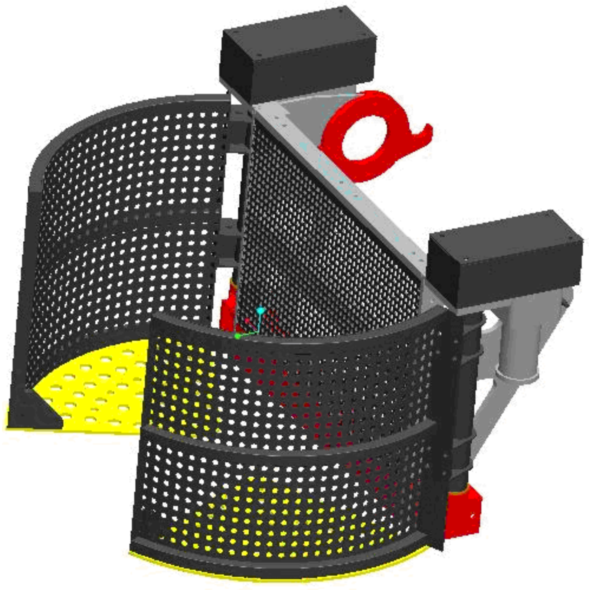


Figure 2.14: DRA Dump Assembly with clamshell system [7]

2.6.2 *DRA Operation*

The DRA was designed to lift typical litter bags of about 4.5 kg (10 lbs) in a reasonable amount of time (cycle time will be discussed in later sections) [7]. The truck must be driven such that the DRA is directly above the litter bag. The unit can then be deployed around the litter, the clamshell closed, and the bag lifted and dumped.

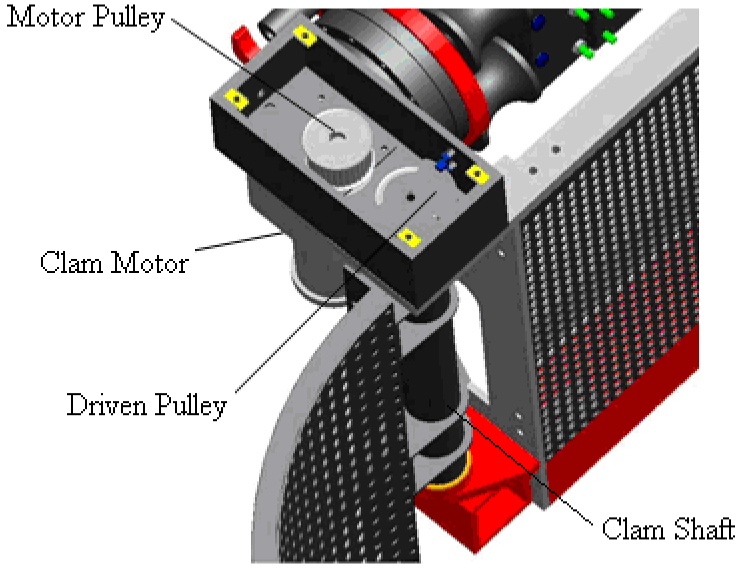


Figure 2.15: Clam Drive Mechanism (Timing belt hidden)

Though it is expected to be used primarily for litter bag collection, the DRA was also designed to lift heavier single item debris such as deer carcasses, tires, chairs, etc., up to 68 kg (150 lbs). This will be accomplished by attaching a fork or platform to the bottom of the dump assembly to assist in lifting larger single item debris. Lifting larger debris will require the operator to come out of the vehicle, affix the fork or platform to the bottom of the DRA shuttle, and place the debris onto the platform. This will save the Caltrans workers from having to lift heavier items into the truck bed themselves. The DRA's collection sequence for a typical litter bag is shown in Figure 2.16.

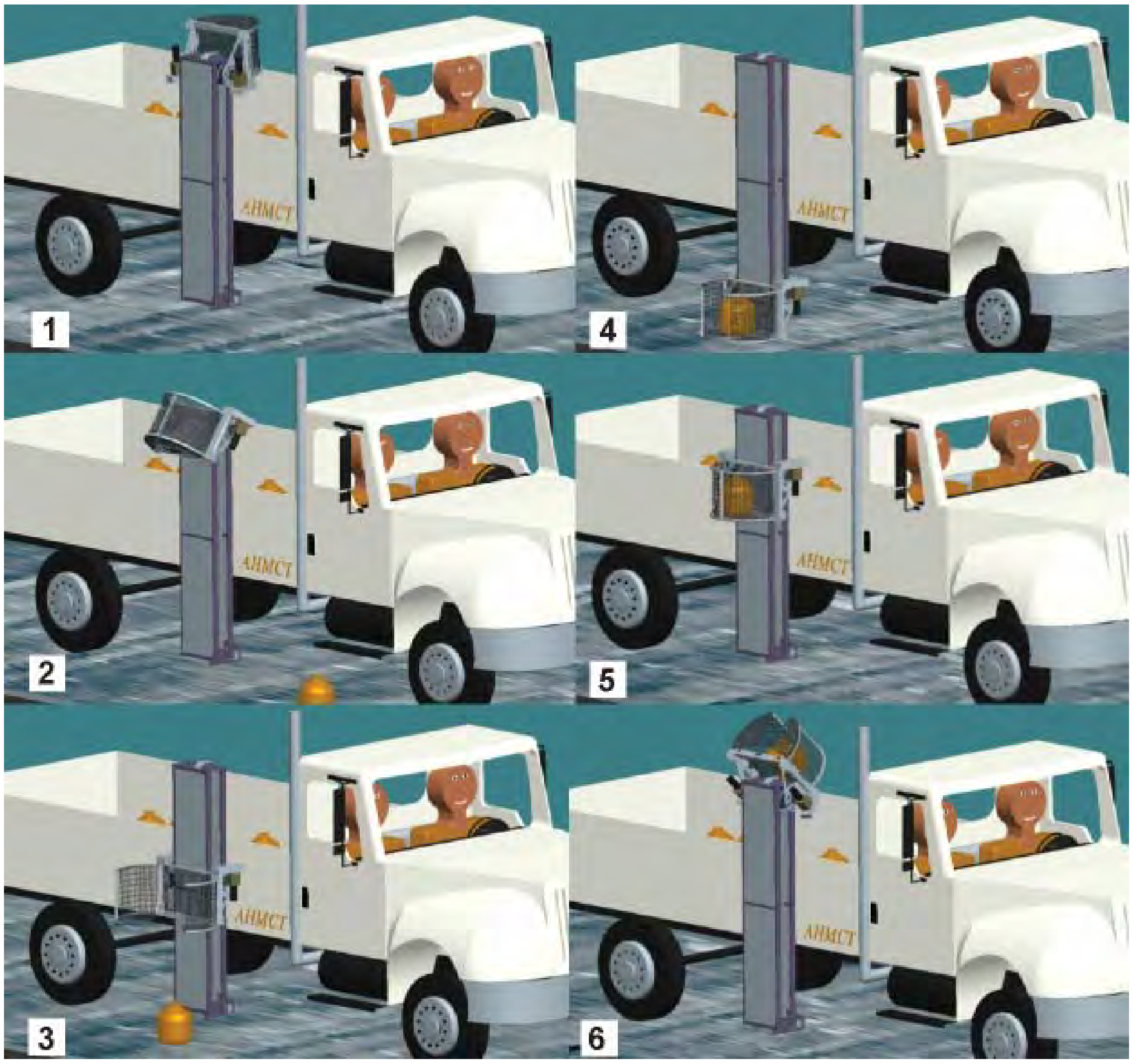


Figure 2.16: DRA collection sequence

With the independent clam system, AHMCT anticipates that the DRA will be able to perform a rolling litter collection at low speeds. With the front most clam open and the rear clam closed, the truck should be able to drive the DRA into a litter bag where it can be scooped and lifted. When traveling and between uses, the DRA can be placed in a docked position. When the dump assembly is rotated to the inside of the truck bed (180^{o}), the shuttle can travel down the frame to the docked position. Screen shot 1 of Figure 2.16 shows the DRA coming out of the docked position.

2.6.3 Advantages and Disadvantages

Overall, the DRA has many advantages, including:

- Workers can remain in truck

- Unit can attach to current vehicles
- Track does not affect truck's current functions
- Inexpensive and no dedicated vehicle
- Simple to learn and operate
- Unit can lift up to 68 kg (150 lbs)
- Quick and effective to deploy
- Possibility for rolling litter bag collection

Some of the disadvantages of the DRA system are:

- Driver must position the DRA directly above the litter bag by steering the vehicle
- When collecting heavier or awkward debris, the worker must still get out of the vehicle to attach lift platform and to place debris

CHAPTER 3: SENSORS, LOGIC CONTROL, AND DRIVE ELECTRONICS

3.1 Introduction

This chapter presents all of the electronic components selected for the DRA. The chapter begins with a description of the requirements for the DRA's electrical system, then goes on to talk about each electrical component used in the system. Each component is described to a level sufficient to understand how the component works, why it was selected over other options, and what purpose it will serve in the overall electrical system. The data sheets for most of the components are located in Appendix D at the end of this report. Though the electronics and motors are presented in separate chapters (Chapters 3 and 4 respectively), selection of the components was a somewhat iterative process.

3.2 General Requirements and Overview

The general requirements of the DRA's electrical system are fairly straightforward. In order to be powered by the vehicle's battery system, the DRA's electrical system must operate at 12 VDC [7]. The electronics need to be able to sense the configuration of the DRA's mechanical components and to restrict and allow specific actuator motions based on that configuration. There needs to be limit sensors to detect the configuration of the DRA, a logic control unit (LCU) to interpret these sensor readings and issue commands, electronic actuators (motors and brakes), and power electronics to supply enough power to properly and sufficiently actuate the DRA. The electronics also need to be able to receive instructions from a human operator and to alert the operator in some way if an error occurs. As will be discussed in more detail in Chapter 3, the rotation and vertical motors each require a brake that can be disabled electronically by a relay. Last, the electronics need to be kept simple to keep cost down and minimize construction and maintenance procedures.

3.3 Limit Sensors

Five limit sensors were used on the DRA to detect the following limits of motion: top of vertical track, rotation at 0°, rotation at 180°, left clam full open, and right clam full open. This is the minimal sensor configuration to sufficiently detect the configuration of the DRA in order to issue motor commands that can not cause a mechanical interference. For example, if the shuttle is in the middle of the track the dump assembly must not be allowed to rotate, as this would cause an immediate mechanical interference. Other sensors that can be incorporated into the DRA include a bottom of vertical track and shuttle docked; this is discussed in more detail in Chapter 6: Conclusions and Recommendations (Section 7.3.3).

The sensors are only required to detect the limits of motion of the DRA and not the absolute or relative position. Non-contact digital magnetic sensors were used for four of the limit sensors because they are inexpensive and robust. Two different magnetic sensors were used, a surface mount digital vane sensor for the clams limit sensors, and a threaded barrel magnetically actuated proximity sensor for the two rotation sensors. A lever type contact switch was used for the top of the vertical track limit.

For the clam limit sensing, series VN1015 digital vane sensors by Cherry Corporation were selected (see Figure 3.1). Sensor VN101503 was selected, which offers a 4.5-24 volt operating range, -40 to 85^{o}C temperature operating range, and 24 AWG x 150mm wire leads [10]. The sensor is magnetically actuated when a ferrous metal is passed between the towers. On top of the clam shaft geared pulley (Figure 2.15) is a thin ferrous disk, which is used as the flag to signal the end of motion of the individual clams. This flag can be adjusted to meet the needs of a particular application. The sensor has three lines - Vin, ground, and the output. It requires a pull-up resistor between the Vin and output pins. When there is a ferrous metal between the towers, the sensor output is 5V (Vin). When the sensor is not activated, the sensor output is 0V. A sufficiently high value pull-up resistor must be used to prevent excessive current from flowing when the output is 0V. The interfacing schematic is shown in Figure 3.2.

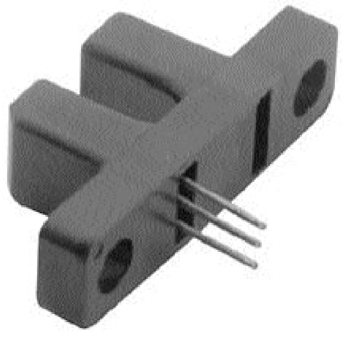


Figure 3.1: VN1015 Cherry Corp. Vane Sensor [10]

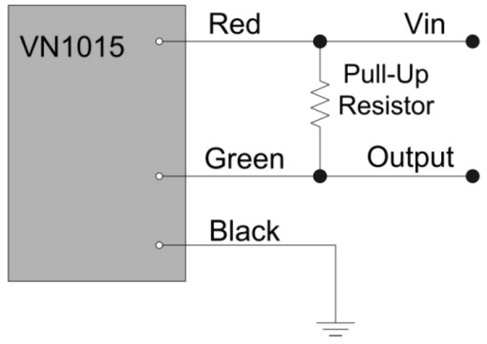


Figure 3.2: VN1015 Sensor Interface

For 0^{o} and 180^{o} rotation limit sensing, Hamlin's 59065 threaded barrel magnetically operated proximity sensors were used (see Figure 3.3) [11]. The sensor functions as a normally open switch that is closed when the magnetic actuator is brought within range. The wiring used for this sensor is shown in Figure 3.4. When the switch is open (not actuated), the pull-up resistor drives the input pin to 5V (Vin). When the magnetic actuator is brought within range, the switch closes (actuated) and the output pin is connected directly to ground. Like with the Cherry vane sensor, a sufficiently high value pull-up resistor must be used to prevent an excess of current from flowing when the switch is closed.

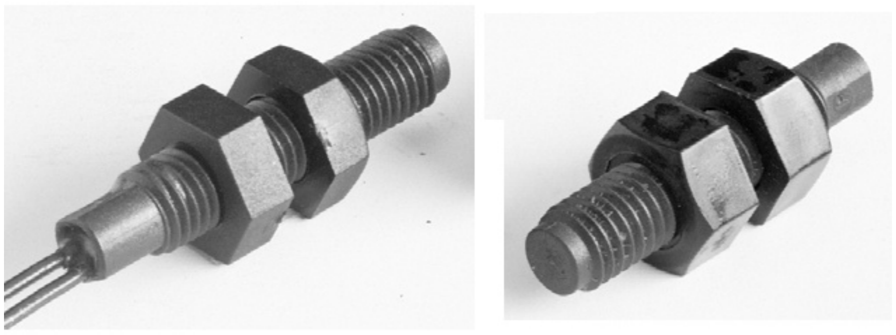


Figure 3.3: a) Hamlin 59065 Proximity Sensor b) Magnetic Actuator [11]

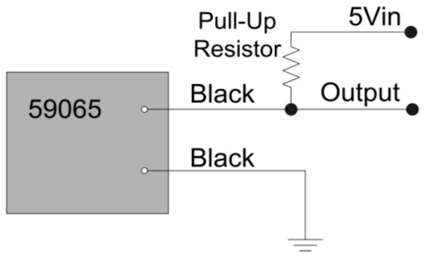


Figure 3.4: Hamlin 59065 Sensor Wiring Interface

The E1101 sensor by Control Products Inc. was used to sense the top of track limit of travel [12]. The E1101 is a normally open angle mounted contact switch, shown in Figure 3.5. The sensor is actuated by pressing a flexible metal lever. The flexibility of the lever allows for some overshoot without compressing and damaging the sensor. The sensor is interfaced the same as the Hamlin 59065.

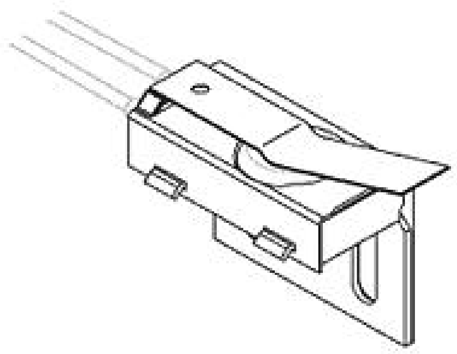


Figure 3.5: Control Products Inc. E1101 Lever Switch

3.4 Motor Controllers

There needed to be a dual axes high power motor controller for the rotation and vertical motors, and a lower powered dual axes motor controller for the two clam motors. The requirements for the high powered motor controller were that it needed to operate at 12 volts, provide sufficiently high continuous current capability, be easy to interface with, and be cost effective. From the motor selection process outlined in Chapter 3, the high power motor controller needed to be able to source a peak current of 44 amps for around 10 seconds for the maximum load. The low power motor controller also needs to operate at 12 volts, provide sufficient current, be easy to interface with, and be cost effective. Again from Chapter 3, the low power motor controller needed to be able to source up to 7.3 amps momentarily.

3.4.1 High Power Motor Controller

The high power motor controller selected was the AX3500 by RoboteQ (see Figure 3.6) [13]. In addition to high power capabilities, the AX3500 has additional intelligent features in a moderately sized board. By performing some intelligent control separate from the LCU, the LUC is freed up for other operations. The AX3500 was selected because it:

- has a high current rating of 60 amps for 30 seconds and 40 amps continuously per channel,
- has protective features such as user preset current limiting, temperature based current limiting, and under voltage shut-down,
- offers two way RS232 serial control, allowing the LCU to send precise motor commands monitor the AX3500's operation,
- it has data logging capabilities useful for the DRA testing phase,

- and it has 4 general purpose analog and digital inputs and 1 digital output (up to 2 amps) for possible future expansion of the total system's input/output capacity
- The AX3500 can be powered with 12-40 volts and is rated to supply 60 amps for 30 seconds or 40 amps continuously.

All design lift times (see section 4.1.2) for the DRA are less than 30 seconds, and 60 amps is a sufficient limit on the current that the motor will be allowed to draw. If a motor attempts to draw too much current due to a stall or a short, the controller automatically limits the current to a user preset value. The AX3500 also automatically limits the current draw of the motors based on the temperature of the MOSFETs (metal oxide semiconductor field effect transistors). This will protect the AX3500 from damage due to overheating in any condition or environment. The AX3500 also features under and over-voltage shutoff protection.

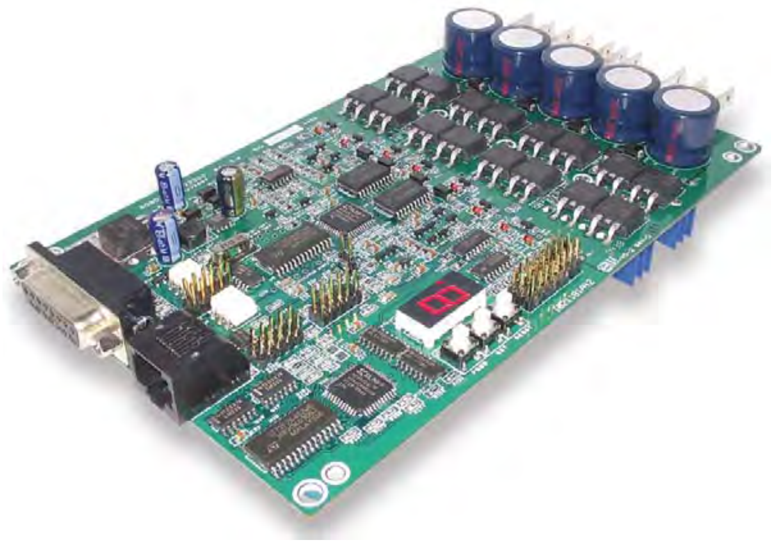


Figure 3.6: AX3500 High Power Motor Controller by RoboteQ [13]

The AX3500 offers three control methods: (1) 0-5 volt analog control, (2) RC pulse protocol control, and (3) two way RS232 serial control. All methods require two communication lines; one for each motor channel. The control method of choice for the DRA is RS232 serial control because it allows for the LCU to access more features of the AX3500, including monitoring its operation, sending precise motor commands, and setting control parameters while in operation. The LCU can read values such as the current levels in the motors, the temperature of the MOSFETs, and the value of any of the general purpose IOs. If needed, the two way communication also allows the LCU to control the high current digital output. The complete IO pins of the AX3500 include one high powered (2A) digital output, two digital inputs, and two analog inputs.

RoboteQ provides computer software, called RoboRun, to program the AX3500's control parameters [14]. Some of the control parameters include individual motor current limits and motor accelerations, as well as a number of other parameters not relevant to the DRA's specific operation or the RS232 control method. These values can also be changed during operation by the LCU. The RoboRun software also allows logging of the AX3500's operation including motor current, MOSFET temperatures, and other parameters, for up to 30 minutes.

The AX3500 can be purchased directly from the manufacturer or through other online robotics and electronics dealers. It is available with either heat sinks on the MOSFETs or a conduction plate for higher current capabilities. The AX3500 with heat sinks was selected, which costs $395.00

3.4.2 Low Power Motor Controller

For the low powered motor controller, the OX2 by Pololu was selected [15]. The OX2 is a dual function board and, as discussed in Section 3.8, will also perform the logic control of the DRA. The OX2 consists of two boards stacked vertically, with the top board serving as the dual channel motor controller and the lower board serving as the logic control unit. There are two motor controllers available with the OX2, the VNH3 and the VNH2. The VNH3 option can source up to 9 amps continuous per channel and does not offer current sensing. The VNH2 can source up to 14 amps continuous per channel and offers current sensing. The VNH2 option was selected for the higher current rating and current sensing capabilities. The rest of the OX2's details are discussed more completely in Section 3.6.

3.5 Human-machine Interface

The human-machine interface needed to be simple and intuitive to operate. It only requires a short range, just enough to be controlled from inside the truck and about 20 feet outside of the truck. It also needed to offer some sort of feedback to alert the user of errors and other problems. A PlayStation® 2 (PS2) wireless controller was selected as the human-machine interface [16, 17]. The PS2 wireless controller was selected because it:

- has sufficient range

- has 16 buttons
- has 2 dual axes proportional joysticks
- has a comfortable and easy to use design
- is small enough to fit in a pocket
- has tactile feedback (vibration)
- is very inexpensive and readily available
- is easy to find online documentation of the interfacing protocols

Four communication lines are required to read the PS2 controllers. This includes a clock line, command line, data line, and attention line. With these four communication lines, all 16 buttons and both dual axis proportional joysticks can be read. Also, vibration commands can be sent to the controller, which can be used to alert the operator of a system error. Most of the buttons on the PS2 controller will be nonfunctional, but they include: four buttons across the top, L1, L2, R1, R2; four directional buttons on the left side face, left, right, up, down; four shape buttons on the right side face, square, triangle, circle, and X; and four in the center of the controller, start, select, L3, and R3.

Two different PS2 wireless controllers as well as one wired PS2 controller were purchased so that their performance and operation could be compared. One wireless controller was the official PS2 controller which was licensed out to a company called Katana [16]. The second controller was the Logitech® Cordless Action™ controller [17]. Both controllers operate at a 2.4 GHz radio frequency, offer vibration feedback, and have a power saving sleep mode. The PS2 controller by Katana and Logitech are shown in Figure 3.7 and Figure 3.8 respectively. Units like these can be purchased for around \$20.00-\$30.00 at most stores that sell video game accessories, and can also be found online. The wired controller also used was the official PS2 controller by Sony, which looks similar to the Katana controller in Figure 3.7.

Figure 3.7: Official PS2 Wireless Controller [16]

Figure 3.8: PS2 Wireless Controller by Logitech [17]

3.6 Brake Relay

To enable and disable the motor brakes, dual relays were used. The power required to switch a relay is too great for a microcontroller to produce, so a high-current relay driver was needed. A dual relay board by Twin Industries was selected, which has a relay driver that can be switched by a low power logic high (+5V) signal from a microprocessor [18]. The board is powered by 12V, and can switch up to 5 amps per channel. The board is shown in Figure 3.9, and can be purchased for around $16.00 at a Fry's electronics store or online.

3.7 Five Volt DC Regulator

The PS2 controller and all of the sensors require a 5V power supply. A switching voltage regulator was used to step down the 12V truck battery to a usable 5V supply. A regulator by Power-One was selected because it fulfilled this requirement and was on-hand at the AHCMT Research Center [19]. The 5V regulator unit is shown in Figure 3.10.

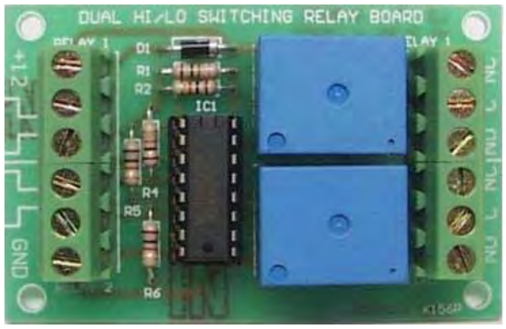


Figure 3.9: Two Channel Relay Board [18]

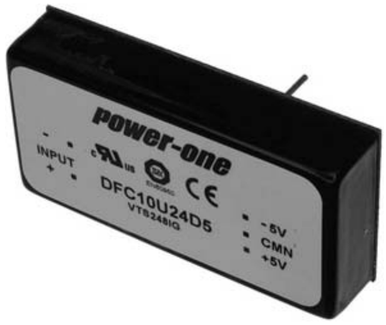


Figure 3.10: 5VDC Regulator by Power-One [19]

3.8 Logic Control Unit

To pull it all together, the logic control unit must be able to monitor the sensors, determine the configuration of the DRA, read the human-machine interface, and then send commands to the motor controllers based on the DRA configuration and operator commands. It also needs to be able to perform some level of automation, and also be able to monitor the DRA's electrical system for errors and alert the operator of the error.

The OX2 by Pololu was selected to serve as the logic control unit for the DRA (see Figure 3.11) [15]. Pololu is an online supplier of a wide variety of robotics electronics including various motor controllers, microcontroller units, servo controllers, sensors, and robotics kits.

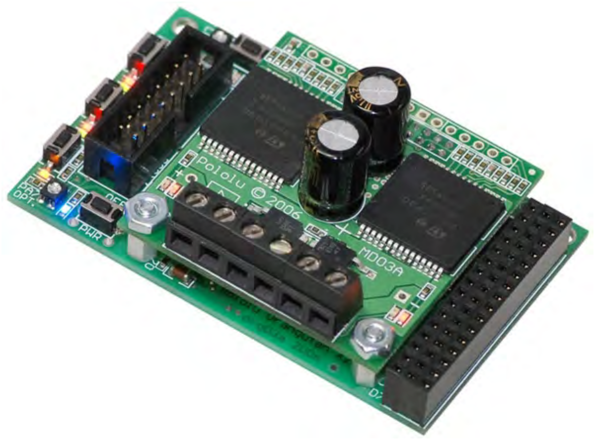


Figure 3.11: Orangutan X2 Robot Controller [15]

The OX2 is what Pololu calls a robot controller because it integrates a microcontroller unit with a dual channel motor controller. The microcontroller will provide the logic control needs while, as mentioned previously, the low powered dual motor controller will be used to power the two clam motors. The OX2 also connects to a 4x20 character LCD (liquid crystal display), which is critical for the development of code and debugging. The OX2 was selected over other options because it:

- has sufficient computational power and input/output (IO) capability for the DRA and its sensors
- has a dual microcontroller architecture which makes it versatile and easily configurable to the DRA's specific requirements
- can provide the logic control and power for two low current electric motors with one compact device

- has a free downloadable software development suite and is programmable in C, a very common, well documented, and easy to use language
- has an easy to use online forum maintained and frequently monitored by Pololu, as well as other unofficial forums dedicated to the AVR microcontrollers which power the OX2
- and it is a low cost component, which keeps costs down and eliminates the need for a separate low power motor controller, further reducing the costs

The OX2 features dual microcontroller architecture with a main and auxiliary microcontroller. Both are 8-bit AVR microcontrollers by Atmel. The auxiliary microcontroller is the ATmega168, which is preprogrammed and cannot be reprogrammed by the user. The auxiliary microcontroller does all of the motor control functions like pulse width modulation (PWM) and motor current sensing, and directly controls the VNH2 motor controller. This frees up the main microcontroller from having to monitor these functions. The ATmega168 receives commands sent by the main microcontroller over its SPI (serial peripheral interface), and can also send information back to the main microcontroller. The auxiliary microcontroller is also connected to a USB port and serves as the programmer for the main microcontroller. Pololu provides a library of functions in C for the communication between the two microcontrollers.

The main microcontroller is an ATmega644 by Atmel [20]. This is the microcontroller where the user's program is loaded and executed. It has an 8-bit RISC (reduced instruction set computer) architecture with 64 Kbytes of flash program memory, 4 Kbytes of SRAM, 2 Kbytes of EEPROM, and 32 IO pins (not all are available to the user). It operates at of 20 MHz and executes most instructions in one clock cycle allowing up to 20 MIPS (million instructions per second) of code execution.

The IO requirements for the DRA are: 4 pins for the PS2 controller, 2 pins for the AX3500, 2 pins for the vertical and rotation brakes, and 7 pins for the limit sensors (including limit sensors to be added later), for a total of 15 needed pins. Of the 32 IO pins on the main microprocessor, 16 are immediately available on the IO bus on the right side of the board, of which 8 can be configured as analog inputs [15]. There are also two unused IO pins available on the bottom of the board, for a total of 20 undedicated IO pins. The remaining IO pins are dedicated to the SPI communication between the main and auxiliary microcontroller and for the LCD display connector. The 20 undedicated pins are more than sufficient for the IO needs of the DRA. There is an LCD option when ordering the board which is directly compatible with this connector. If the LCD is not needed, the LCD connector pins can be used as additional general purpose IO pins. The OX2 also has internal pull-up resistors (20-50 kΩ) that can be activated on each IO port. This eliminated the need for an external pull-up resistor on the sensors.

The OX2 has a two-board design. The top board contains the motor drivers and power terminals, while the bottom board contains the dual microcontrollers, supporting electronics, and IO bus. This allows the OX2 to have a footprint smaller than a credit card while still offering considerable computational and motor control power. Some important components on the top board are shown in Figure 3.12 and some important features of the bottom board are shown in Figure 3.13. The motor control capabilities of the OX2 were discussed in more detail in Section 3.4.2.

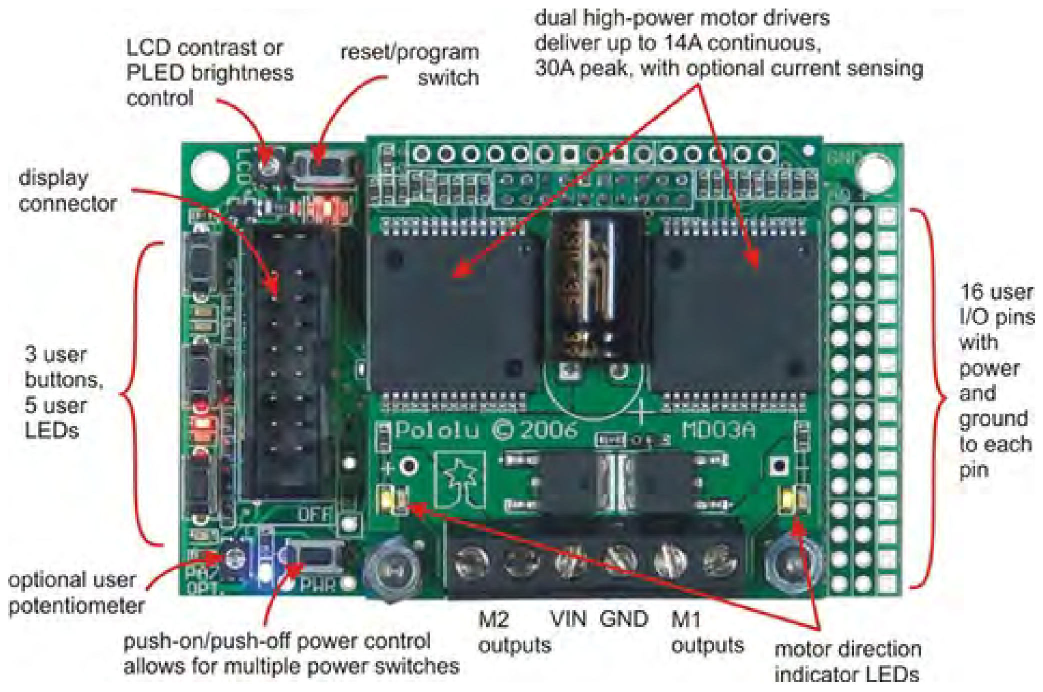

LCD contrast or PLED brightness control
reset/program switch
dual high-power motor drivers deliver up to 14A continuous, 30A peak, with optional current sensing
display connector
3 user buttons, 5 user LEDs
16 user I/O pins with power and ground to each pin
optional user potentiometer
push-on/push-off power control allows for multiple power switches
M2 outputs
VIN
GND
M1 outputs
motor direction indicator LEDs
Pololu © 2006
MD03A
OFF
PWR

Figure 3.12: OX2 Top View [15]

Atmel has a free software call AVR Studio 4 which allows programming and debugging of their microcontrollers in their native assembly language [21]. The ATmega644 can also be programmed in C through a third party software suite of executables called WinAVR [22]. WinAVR is an open source set of software development tools for Atmel's AVR microprocessors, which can be integrated directly into AVR Studio 4. Pololu provides a library of easy to use functions written in C for operating the LCD and motor controllers for the OX2.

The ability to program the OX2 in C and these provided functions greatly simplifies programming the OX2.

The Pololu website (pololu.com) has a forum for their products, including the OX2 [23]. The forum is monitored by Pololu's engineers who provide advice and answer questions that are posted to the forum, typically within a few hours of the post. They also provide help for the entire spectrum of robotics projects: programming, sensors, actuators, electronics, etc. This service was used several times and proved to be a valuable troubleshooting resource for the OX2 and code development. There is also an unofficial third party forum dedicated to AVR microcontrollers, called AVR Freaks [24]. This forum is more a general source for AVR microcontrollers and AVR Studio 4.

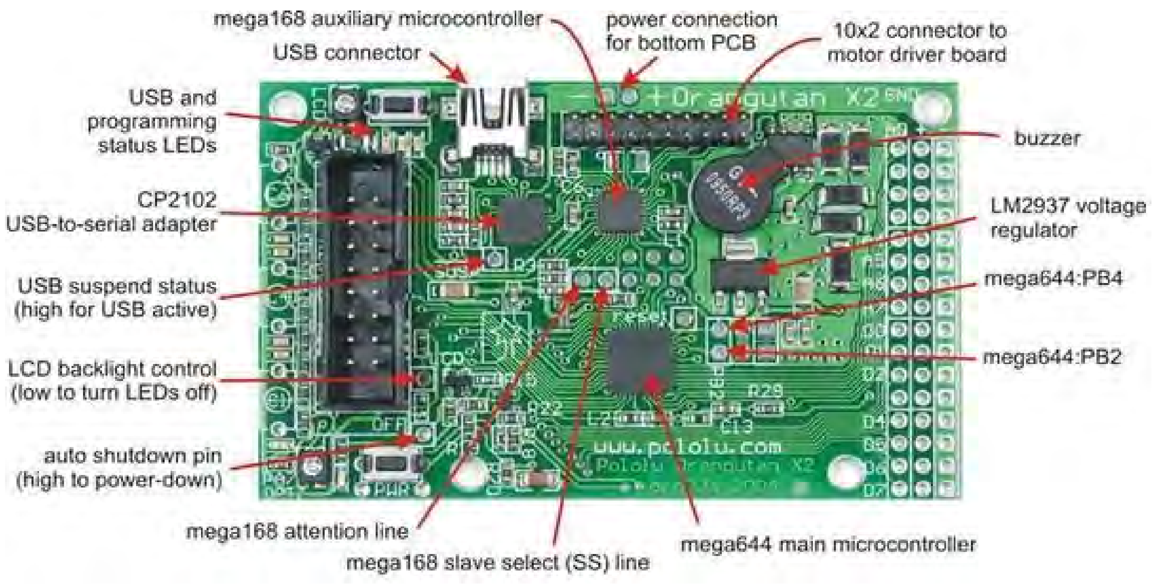

mega168 auxiliary microcontroller
USB connector
power connection for bottom PCB
10x2 connector to motor driver board
USB and programming status LEDs
buzzer
CP2102 USB-to-serial adapter
LM2937 voltage regulator
USB suspend status (high for USB active)
mega644:PB4
mega644:PB2
LCD backlight control (low to turn LEDs off)
auto shutdown pin (high to power-down)
mega168 attention line
mega168 slave select (SS) line
mega644 main microcontroller

Figure 3.13: OX2 Lower Board [15]

The OX2 provides a low cost solution for the DRA logic control needs. The board itself costs $119.00 with the VNH2 motor controller option and the 20x4 character LCD screen cost $30.00. The LCD was critical in the code development and debugging processes. The OX2 can be ordered through Pololu directly, or through other suppliers. The OX2 also reduces costs by eliminating the need to purchase a logic control unit and low power motor controller separately. Research showed that purchasing a separate equally powered motor controller and similar microcontroller logic control unit would cost approximately the same individually as the OX2 does for the combined feature.

CHAPTER 4: MOTOR SELECTIONS AND PERFORMANCE CALCULATIONS

4.1 Introduction

For reasons stated previously, electric motors were the choice actuators for the DRA. In order to select the most appropriate motors, it was important to understand the effect of the motor characteristics on the lift cycle times of the DRA. Time was spent developing equations to estimate the lift times of the two critical drives- the vertical and the rotation drives- so that the lift times could be evaluated. Approximations were used to simplify the system equations to a level sufficient to estimate the lift times. In the case of the vertical drive, an analytical expression of performance was obtained, and in the case of the rotation drive, a simulation ready differential equation was obtained. Performance constraints (which are determined in Section 4.1.2) were applied to narrow down the available motors, and the final motors were ranked based on a selection objective. For the less-critical clam motors, a much simpler selection procedure was used based on the closing time and maximum closing force of the motor.

Note that in the following analyses, all equations derived assume inputs and outputs of SI units. Subsequently all parameters taken from the motor data sheets were first converted to SI units. Note also that speed and torque of the gearbox refers to the output shaft of the gearbox, and speed and torque of the electric motor refers to the output shaft of the motor before the gearbox if one is present.

4.1.1 Motor Vendor Selection

Groschopp Inc. was selected as the vendor for all of the motors for several reasons: (1) they have a wide selection of 12 volt permanent magnet motors over a range of power outputs and speeds; (2) they have easy online access to a detailed performance sheet and CAD file for each of their motors; and (3) they have several brake options for each motor [25]. An example of a typical motor data sheet is shown in Figure 4.1.

MOTORTEC™ DATA SHEET

Prepared by: Groschopp.com 01/06/09

Motor:	PM 10818	
Voltage:	12 v DC	
Speed:	1100 rpm	2 Poles
Gearbox:	PL 7300	20:1 Ratio

Product ID: 64503

SPECIFICATIONS

Rating:

Speed:	55.8 rpm
Torque:	169.3lb-in
Current:	15.37 amps
Output:	112 watts
Output:	0.1499 HP

Duty Cycle:

On:	Continuous
Off:	

Efficiency:

Gearbox:	87.7 %
Motor:	68.8 %
System:	60.3 %

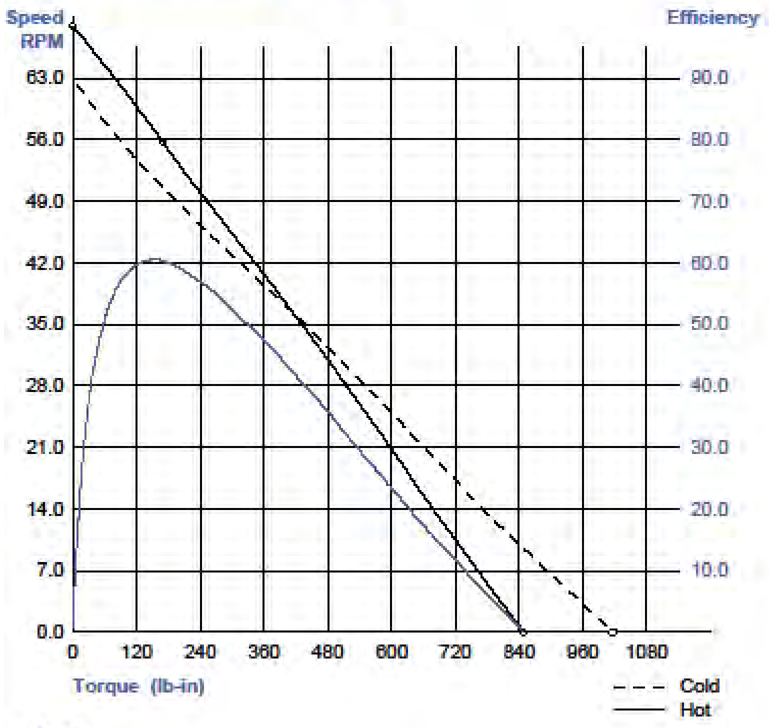
Start/Stall Conditions:

Current:	79.24 amps
Torque:	1017.21 lb-in

Constants:

Ke:	9.01 v/krpm
Kt:	0.76 lb-in/amp

Speed, Efficiency Vs. Torque



Notes:

Figure 4.1: Typical Groschopp Motor Data Sheet [25]

4.1.2 *Motor Selection Constraints and Objectives*

In order to be attractive as a practical alternative, the shuttle must be able to lift a typical load of 4.5 kg (10 lbs) within a reasonable timeframe. For a typical load, the dump cycle design time was 7-12 seconds. This was broken down as 4 to 7 seconds for the vertical path, 2-3 seconds for the rotation, and 1-2 seconds for the clam closing operation. The lower limits are imposed in order to keep the speed of the system from becoming too large such that collisions and high momentum might damage the system.

The lift time for the maximum load of 68 kg (150 lbs) must also be within a reasonable limit. For the maximum load, the dump cycle design time was 30 seconds or less. This was broken down as 20 seconds or less for the vertical path and 10 seconds or less for the rotation. For heavier loads, the forks will be used instead of the clams, so no time constraint for the clams is imposed.

One critical requirement of the vertical drive is the ability to stop the shuttle in place. This is particularly important when the shuttle is at the top of its path and begins the rotation process, where because of obvious geometric constraints, the shuttle cannot be allowed to slide down. Therefore another constraint for the vertical gear motor is that the brake and motor combination be sufficient to hold the shuttle in place with the maximum payload.

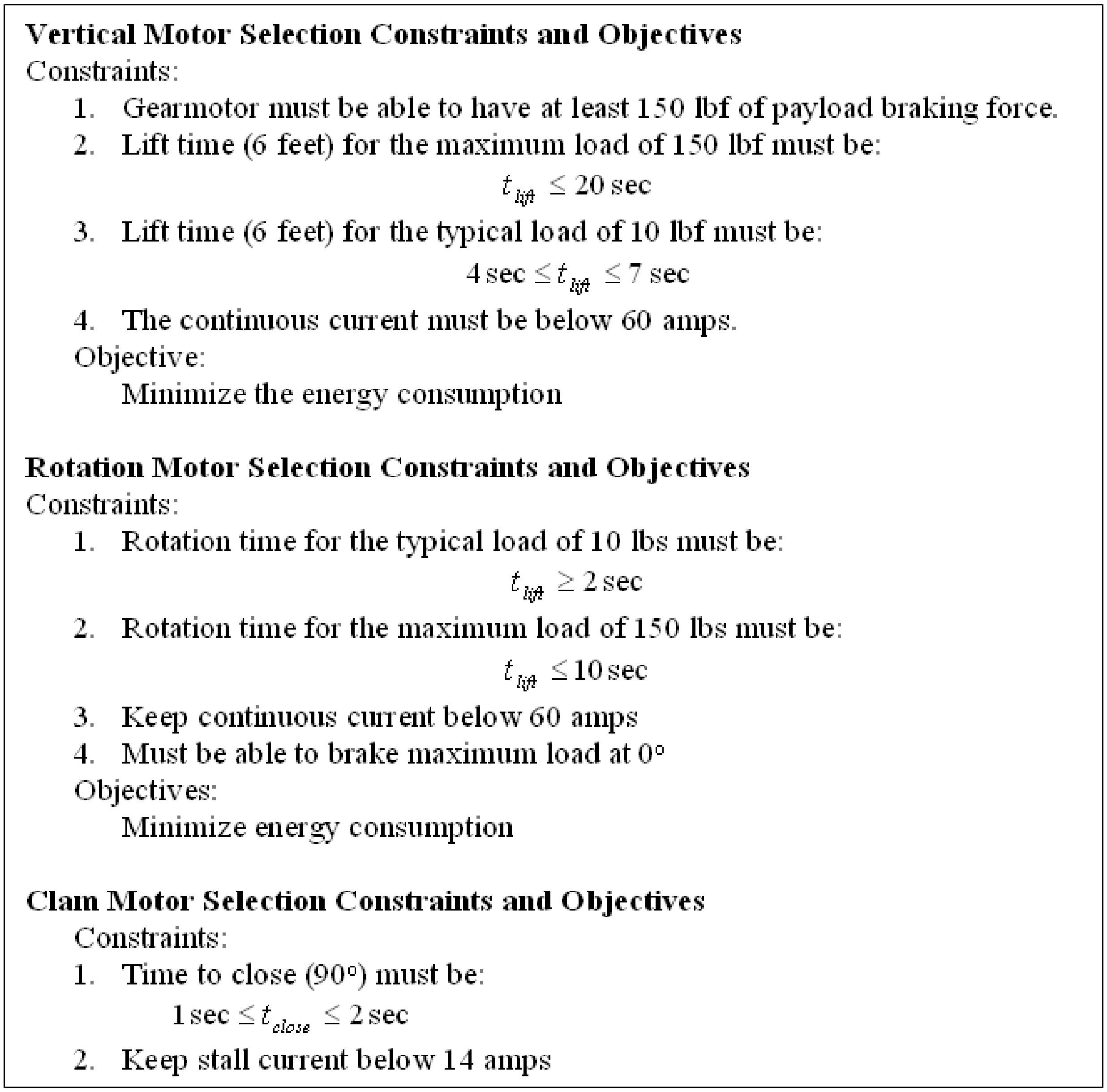
The rotation drive also needs the ability to brake. When the shuttle is in the deployed position (0^o position), the motor must be able to hold the dump frame in place. It is not critical that the brake be sufficient to hold the dump frame in place at intermediate angles, since it is expected that the operator will have no need to stop mid-rotation. Therefore another constraint for the rotation drive is that the motor and gearbox combination be sufficient to hold the shuttle in place with the maximum payload.

The clam motors are not expected to grip onto a litter bag in order to lift it, but rather to scoop the clams under the bottom of the bag. For this reason, it was expected that the less powerful of Groschopp's motors would be sufficiently strong to accomplish this and no strength constraints were placed on the clam motors.

Constraints are also imposed by the limits of the drive electronics. The high power motor controller, which will power the vertical and rotational motors, has a continuous current rating of 60 amps per channel. The low power motor controller which will power the two clams has a continuous current rating of 14 amps per channel. It is important that the motors do not draw in excess of these current limitations.

After the available motors are reduced by the constraints, a selection objective must give a method of rating the remaining motors. One goal of the DRA is to attach to existing Caltrans vehicles with minimal modifications. It is therefore important that the DRA impede as little as possible on the vehicle's systems. For this reason, the objective for the vertical and rotational motor selection is to minimize the energy consumption for a lift cycle.

All of the constraints and objectives for the vertical drive, rotational drive, and clams are summarized in Figure 4.2.



Vertical Motor Selection Constraints and Objectives

Constraints:

1. Gearmotor must be able to have at least 150 lbf of payload braking force.
2. Lift time (6 feet) for the maximum load of 150 lbf must be:
$$t_{lift} \leq 20 \text{ sec}$$
3. Lift time (6 feet) for the typical load of 10 lbf must be:
$$4 \text{ sec} \leq t_{lift} \leq 7 \text{ sec}$$
4. The continuous current must be below 60 amps.

Objective:

Minimize the energy consumption

Rotation Motor Selection Constraints and Objectives

Constraints:

1. Rotation time for the typical load of 10 lbs must be:
$$t_{lift} \geq 2 \text{ sec}$$
2. Rotation time for the maximum load of 150 lbs must be:
$$t_{lift} \leq 10 \text{ sec}$$
3. Keep continuous current below 60 amps
4. Must be able to brake maximum load at 0°

Objectives:

Minimize energy consumption

Clam Motor Selection Constraints and Objectives

Constraints:

1. Time to close (90°) must be:
$$1 \text{ sec} \leq t_{close} \leq 2 \text{ sec}$$
2. Keep stall current below 14 amps

Figure 4.2: Motor Selection Criteria

4.2 Vertical Gearmotor Selection

The vertical motor drives the timing belt which lifts and lowers the shuttle of the DRA. This is a critical component of the system and had to be selected with care. The motor must be able to lift a typical load with reasonable time, while still maintaining the ability to lift a larger load on occasion. This is tricky, however, since the low voltage system for an automotive system imposes power limitations. Because of this power limitation, a drop in one lift time typically means a raise in the other. The right balance between the two times must be found in order to ensure the DRA is quick enough to be attractive for lifting litter bags, while still being able to lift heavier loads with reasonable speed.

The current method for getting deer carcasses off of the roadside is by human labor. AHMCT believes that any alternative method presented is expected to be an attractive substitute as long as the process does not take an unreasonable amount of time. Considering this, more emphasis will be placed on keeping the typical load lift times down over lowering the maximum load lift times.

4.2.1 Derivation of Mechanical Performance Equations

Derivation was done by using Newton's Law, $F = ma$. Figure 4.3(a) is a diagram of the vertical drive system and Figure 4.3(b) is the free body diagram of the payload (for dynamics reference, see [26]). Ignoring friction effects and assuming that the belt and pulleys are massless, the sum of the forces vertically along the belt (X direction) gives:

$$\Sigma F_X = {}^{\tau_g}\!/_{r} - mg = m\ddot{X} \tag{4.1}$$

where τ_g is the torque applied by the gearbox, r is the radius of the motor gear, m is the total mass of the load (consisting of the mass of the shuttle m_s and the mass of the payload m_{pl}), and g is the acceleration of gravity. This model ignores friction mainly because friction levels could not easily be estimated for the shuttle translation due to too many unknown factors. It is expected that this will be a reasonable assumption because idle pulleys and timing belts are very efficient and a high degree of accuracy of the vertical lift time is not required.

The torque applied by an electric motor (τ_m) can be approximated as a straight line from the stall torque (τ_o) in Nm to the no-load speed (ω_o) in rad/sec (see Figure 4.4).

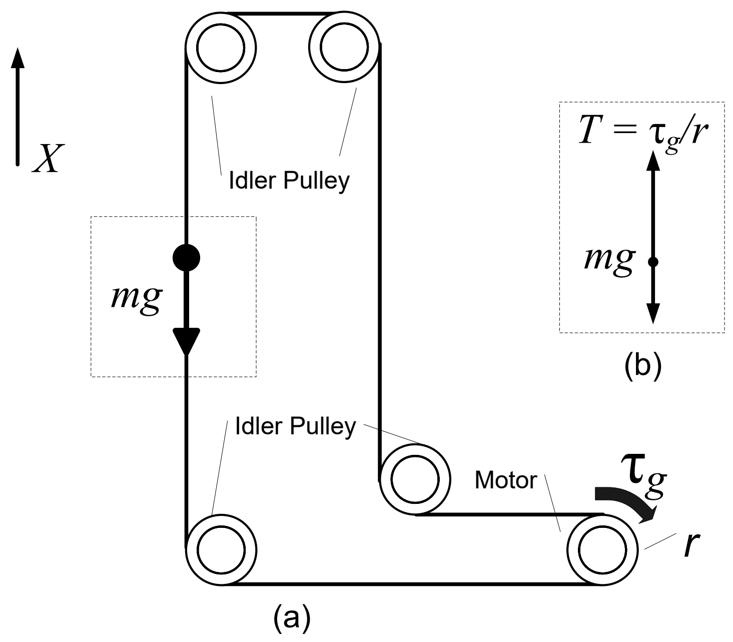


Figure 4.3: (a) Vertical Motor Pulley Assembly; (b) Free Body Diagram

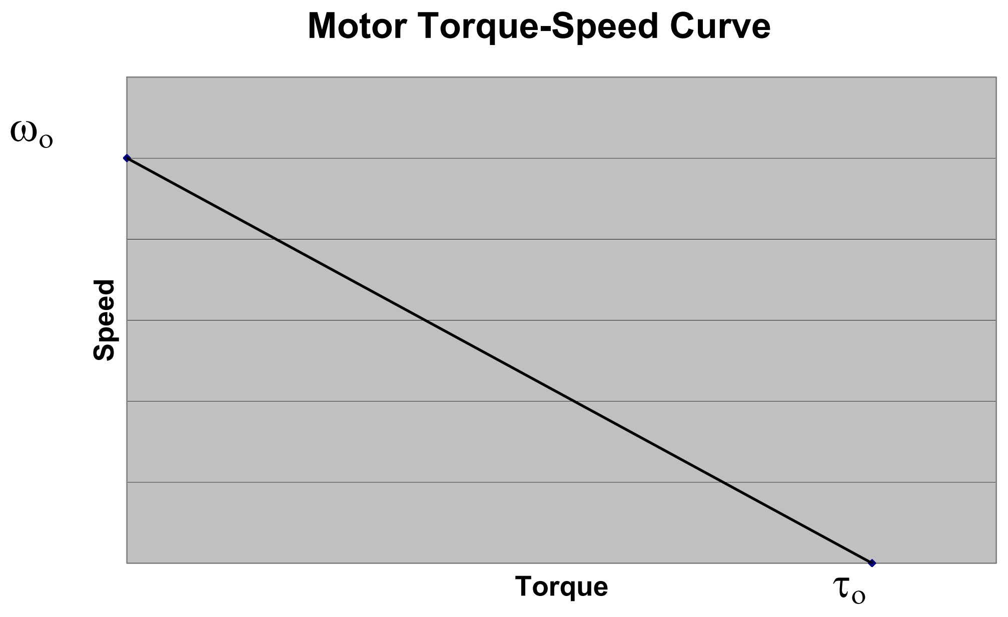


Figure 4.4: Example Speed-Torque Curve

The motor torque can therefore be written as

$$\tau_m = \tau_o - \frac{\tau_o}{\omega_o}\omega_m \tag{4.2}$$

where ω_m is the angular velocity of the motor. The data sheets for Groschopp's gearmotors gives the stall torque and no load speed output after the gearbox, so in this case, equation (4.2) represents the torque output after the gearbox ($\tau_m = \tau_g$) with ω_m equal to the gearbox speed ω_g. If $\omega_g = \dot{X}/r$ is substituted into (4.2), equation (4.1) becomes

$$m\ddot{X} = \left(\frac{\tau_o}{r} - \frac{\tau_o}{\omega_o r^2} \dot{X} \right) - mg , \tag{4.3}$$

where τ_o and ω_o are now the stall torque and no-load speeds after the gearbox. Rearranging (4.3) and recognizing that $\dot{X} = V$ and $\ddot{X} = \dot{V}$, where V is the velocity of the timing belt, equation (4.3) becomes

$$\dot{V} + \frac{\tau_o}{\omega_o m r^2} V = \frac{\tau_o}{mr} - g . \tag{4.4}$$

Equation (4.4) is the system equation representing the motion of the vertical drive system. This equation can be solved for V, which can then be integrated to get X.

The homogeneous form of equation of (4.4) is [27]

$$\dot{V} + \frac{\tau_o}{\omega_o m r^2} V = 0 . \tag{4.5}$$

Assume (4.5) has a solution in the form $V_o = A \exp\left(-t/\alpha\right)$, where the subscript "o" represents the homogeneous solution, A is a constant, and α is the time constant of the exponential decay. Substituting the solution into (4.5) gives

$$\frac{\tau_o}{\omega_o m r^2} A \exp\left(-t/\alpha\right) - \frac{A}{\alpha} \exp\left(-t/\alpha\right) = 0 . \tag{4.6}$$

The exponential terms and A are not zero and can be divided out. Solving for the time constant α results in

$$\alpha = \frac{\omega_o m r^2}{\tau_o} . \tag{4.7}$$

For the particular solution to (4.4), assume a solution of the form $V_p = C$, where the subscript "p" represents the particular solution and C is a constant. Substitution of the particular solution into (4.4) gives

$$\frac{\tau_o}{\omega_o m r^2} C = \frac{\tau_o}{mr} - g$$

$$C = \frac{\omega_o r^2}{\tau_o} \left(\frac{\tau_o}{r} - mg \right) . \tag{4.8}$$

The total solution of equation (4.4) is the sum of the homogenous and particular solutions. The total solution is therefore

$$V = V_o + V_P = A\exp\left(-\frac{\tau_o}{\omega_o m r^2}t\right) + \left(\frac{\tau_o}{rm} - g\right)\frac{\omega_o m r^2}{\tau_o}. \qquad (4.9)$$

To simplify this, (4.9) can be written in terms of the constants α and C above, which results in

$$V = A\exp\left(-t/\alpha\right) + C. \qquad (4.10)$$

The solution of interest is when the motor starts from rest at time zero. The initial condition of the motor is therefore $V(0) = 0$. Applying this initial condition to equation (4.10) results in

$$V(0) = 0 = A\exp(0) + C,$$

where is it found that $A = -C$. With this, equation (4.10) becomes

$$V = C\left[1 - \exp\left(-t/\alpha\right)\right]. \qquad (4.11)$$

As time goes to infinity, the exponential decay of equation (4.11) goes to 0, and $V(\infty)=C$. C is therefore the maximum speed (C = $V_{\max}$) obtained by the motor. $V_{\max}$ will have a positive value if

$$\tau_o/r > mg.$$

Equation (4.11) can be integrated to get the displacement of the motor system. Integration results in

$$X = V_{\max}\left[t + \alpha\exp\left(-t/\alpha\right)\right] + B, \qquad (4.12)$$

where B is a constant of integration. Using the initial condition $X(0) = 0$, the final expression for the displacement as a function of time is obtained

$$X = V_{\max}\left\{t - \alpha\left[1 - \exp\left(-t/\alpha\right)\right]\right\} \qquad (4.13)$$

and recall that

$$\alpha = \frac{\omega_o m r^2}{\tau_o}$$

$$V_{\max} = \frac{\omega_o r^2}{\tau_o}\left(\frac{\tau_o}{r} - mg\right).$$

It is difficult to solve equation (4.13) for a closed form expression of time, so the equation was approximated. Equations (4.11) and (4.13) can be approximated with the transient portion removed if the time constant α is sufficiently small compared to the expected lift times. The average time constant, α, of the available motors for the maximum payload was calculated to be 0.0092 seconds with a standard deviation of 0.0145 seconds. Most of the motors were on the low side of the average time constant, with a few outlier time constants on the high side. It can be assumed that the system has reached steady state after 5 time constants, and the average steady state time is therefore 0.046 seconds with a standard deviation of 0.073. Since this is much less than the design travel times (Figure 4.2), the transient part of (4.11) and (4.13) can be neglected, resulting in the approximation equations for velocity and displacement

$$V = V_{\max} \tag{4.14}$$

$$X = V_{\max} t \,. \tag{4.15}$$

These equations are equivalent to assuming that the motor reaches steady state instantaneously. The lift time can be estimated from Equation (4.15) by setting X to the total length of the shuttle track (L_s) and solving for t_{lift}. The lift time is

$$t_{lift} = L_s / V_{\max} \tag{4.16}$$

Equation (4.16) will be used to estimate the DRA lift time.

4.2.2 Derivation of Electrical Performance Equations

The wiring of an electric motor can be modeled by the circuit shown in Figure 4.5. Summing the voltages across the motor terminals results in [28]

$$V_a = K_e \omega + iR_w + L_w \frac{di}{dt}, \tag{4.17}$$

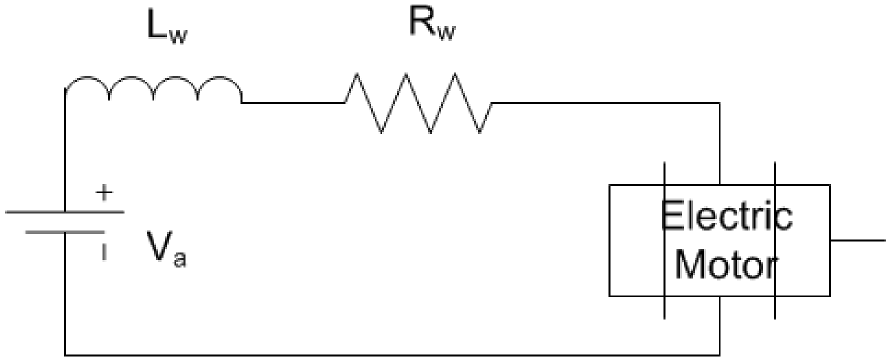


Figure 4.5: Electric Motor Circuit Model

where K_e is the electrical motor constant in V-s/rad, ω is the speed of the motor in rad/s, i is the current, R_w is the winding resistance of the motor, and L_w is the inductance of the motor windings. The first term in Equation (4.17) is the back EMF generated by the motor spinning, the second term is the voltage across the motor windings, and the third term is the voltage due to the inductance of the motor windings. These terms add together to equal the voltage applied across the motor terminals. The inductance of the motor windings is very low and therefore the current reaches steady state very quickly. It is therefore assumed that the motor current reaches steady state instantly and the inductance can be neglected. Setting $L_w = 0$ and solving for the current results in

$$i = \frac{V_a - K_e \omega_m}{R_w} \,. \tag{4.18}$$

The winding resistance can be calculated by $R_w = {}^{V_a}\!/_{i_{stall}}$, and equation (4.18) becomes

$$i = i_{stall}\left(1 - \frac{K_e \omega_m}{V_a}\right). \tag{4.19}$$

Equation (4.19) will allow the evaluation of the motor current.

The objective for the motor selection is to minimize the energy consumption. Energy is the time integral of power, where the power used by the motor is given by $P(t) = i(t)V_a$. The equation for energy is therefore

$$E = \int P dt = \int i V_a dt \,. \tag{4.20}$$

Since it is assumed that the motor reaches steady state in a negligible amount of time, the current can be assumed constant and removed from the integral. The energy used becomes

$$E = iV_a \int dt = iV_a t_{lift} \,, \tag{4.21}$$

where t_{lift} is the total lift time, i is the steady state current given by Equation (4.19), and V_a is the applied voltage. Equation (4.21) can be used rank the final motors based on energy use, which will allow the best motor to be selected.

4.3 Narrowing of Vertical Motor Selection

The constraints in Figure 4.2 can now be applied to reduce the number of potential motors. The motor specifications were taken from the data sheets available on Groschopp's website (see Figure 4.1). Note that in Figure 4.1 there are two speed-torque curves, one for cold and one for hot operation. To get the best representation of the motor performance, the average of the hot and cold curves was used in all calculations. Through Pro/Engineer the mass of the shuttle (m_s) was estimated to be approximately 45kg (100lb). As stated previously, the mass terms in Equation (4.16) will include the nominal mass of the shuttle and the mass of the payload.

Constraint 1: Motor must have 150 lbf of payload braking force

There are 4 available motor brakes: 0.339, 0.565, 1.69, and 3.95 Nm (3, 5, 15, and 35 in-lbs respectively) of braking torque [25]. The brakes are applied before the gearbox, directly on the motor shaft, so the braking torque is multiplied by the gear ratio. The available payload braking force is then

$$BrakeForce = \frac{N * BrakeTorque}{r} - m_S g\,, \tag{4.22}$$

where N is the gear ratio and m_s is the mass of the shuttle. Solving for the gear ratio and inserting the 4 braking torques and required brake force gives the minimum gear ratio in order for each brake to be adequate. These values are summarized in Table 4-1 (rounded up to the nearest whole number).

Braking Torque	Minimum Gear Ratio (N)
0.339 Nm (3 in-lb)	167
0.565 Nm (5 in-lbs)	100
1.69 Nm (15 in-lbs)	33
3.95 Nm (35 in-lbs)	14

Table 4-1: Minimum Gear Ratios for Given Braking Torques

The 35 in-lbs brake is only available on the larger 108XX series motors, while the other 3 brakes are available the 108XX series and on the two smaller series motors, 60XX and 80XX. This means for a gearmotor to be viable for the vertical drive, any 108XX series motor must have a 15:1 or greater gear ratio, and the other two series gearmotors must have a 34:1 or greater gear ratio. These gearbox constraints were used to filter the 85 initial gearmotors down to 54.

Constraint 2: Lift time for the max load must be less than 20 sec

Equation (4.15) was used to solve for the lift time of the remaining motors given the max payload. Five of the 54 motors had negative V_{max}, meaning they could not lift the maximum payload and were eliminated immediately. At this point the 20 second time limit was applied, and the remaining 49 motors were narrowed to 15.

Constraint 3: Lift time for the typical load must be between 4 and 7 sec

V_{max} was recalculated for the typical payload and used in equation (4.15) to estimate the lift time for a typical load. Applying the criteria of $4\,\text{sec} \le t_{lift} \le 7\,\text{sec}$ further narrowed the 15 motors down to 3.

Constraint 4: The continuous current must be below 60 amps

Equation (4.19) was used to calculate the steady state current for the three remaining motors. None of the motors exceeded 60 amps in steady state; none were eliminated.

4.3.1 Final Selection of the Vertical Motor

Equation (4.21) was used to calculate the energy used by the 3 remaining motors for a lift cycle, and is shown in Table 4-2 for the typical and maximum payloads.

Motor Number	6.5 kg		68 kg	
	Energy (J)	Power (W)	Energy (J)	Power (W)
63505	1619	242	5284	525
64502	1798	312	9469	685
64503	1785	254	6157	529

Table 4-2: Energy and Power Used by Final Vertical Motors for a Lift Cycle

Motor 63505 uses less energy for both payloads, and by the objective established in Figure 4.2, is the motor of choice for the vertical drive. Motor 63505 has a 40:1 gear ratio, and therefore was selected with the 15 in-lb brake. It is noted that motor 63505 also minimized the power required for the two load cases.

4.3.2 Performance Calculations for Motor 63505

Excel was used to calculate and plot performance values for motor 63505. Figure 4.6 shows the calculated lift time for payloads from 0 to 70 kg based on Equation (4.16).

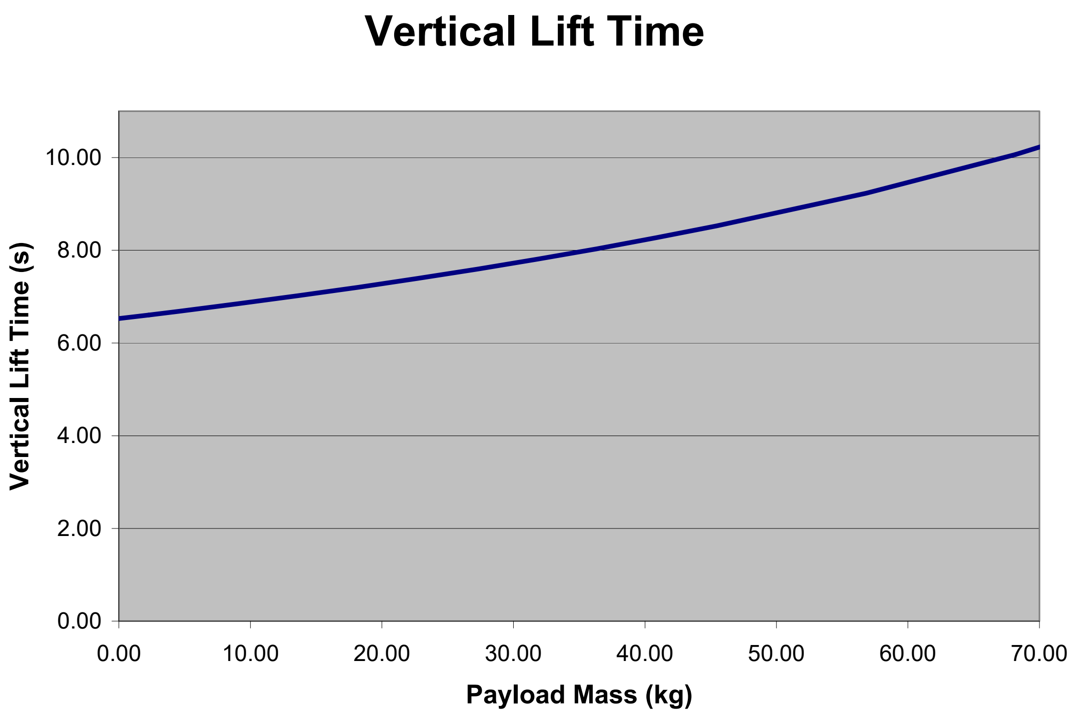


Figure 4.6: Lift times for payloads from 0 to 70 kg

Since the motor is in steady state while lifting the payload, the torque produced by the motor is equal to the torque applied to the motor. The torque produced by the motor (τ_m) can then be calculated $\tau_m = mgr$ (where m is the total mass of the shuttle m_s and payload m_{pl}). The power output of the motor is $\tau_m\omega_m$, and the power input is iV_a. The efficiency is calculated by the ratio of the power output to the power input. These power characteristics for motor 63505 are shown in Figure 4.7 for payloads up to 70 kg. The current, efficiency, and motor torque are plotted on the right axis, while power in and power out are plotted on the left axis.

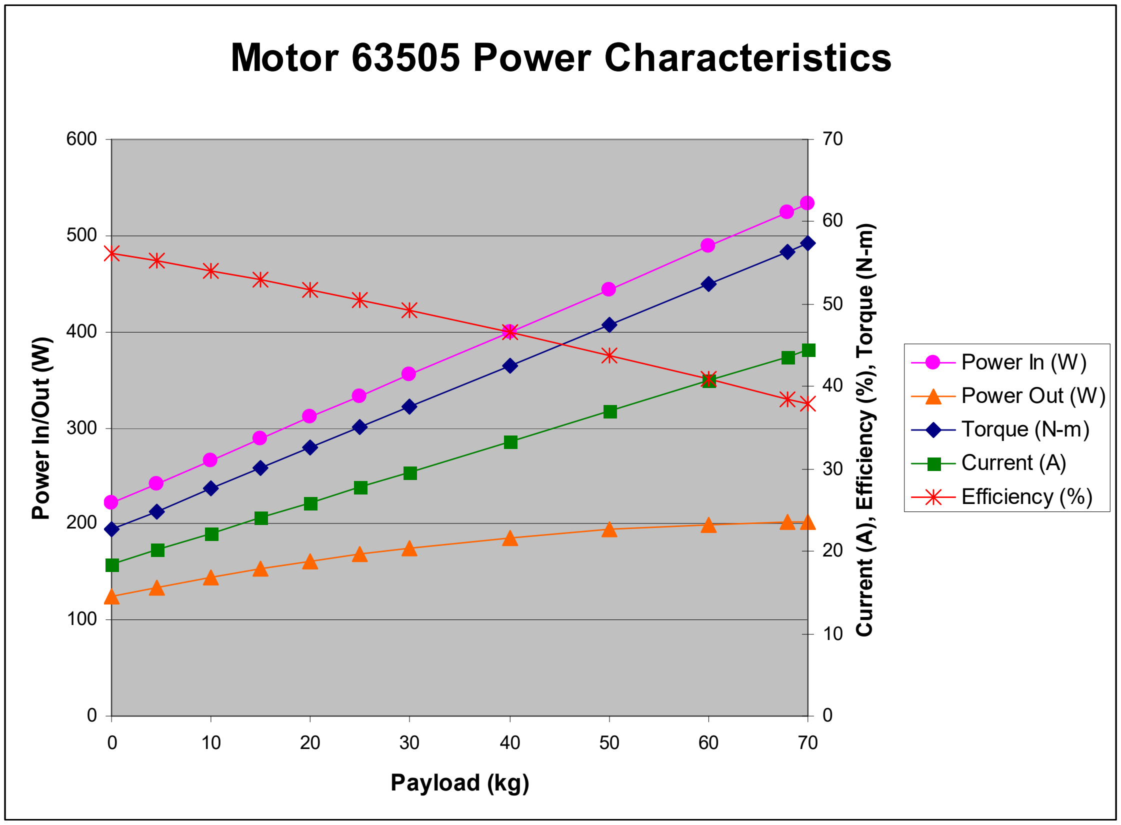


Figure 4.7: Plot of Motor 63505 Power Characteristics

Some of the more important data are summarized in Table 4-3 below with the typical and maximum loads highlighted.

Payload (kg)	Lift Time (s)	Efficiency (%)	Power In (W)
0	6.5	56.1	222
4.5	6.7	55.3	242
10	6.9	54.1	266
15	7.1	53.0	289
20	7.3	51.8	311
25	7.5	50.5	333
30	7.7	49.2	355
40	8.2	46.5	400
50	8.8	43.7	444
60	9.5	40.9	489
68	10.1	38.5	524
70	10.2	38.0	533

Table 4-3: Important Motor 63505 Performance Values

4.4 Rotation Motor and Gearbox Selection

The rotation drive rotates the dump assembly so that the DRA may dump its payload into the bed of the vehicle. The gearboxes provided by Groschopp are too large to fit in the limited space of the DRA shuttle, so an alternative style gearbox was needed. Selecting the rotational drive involved selecting separate motor and gearbox components together. The constraints of Section 4.1.2 were applied to the joint performance of the motor and gearbox configuration.

4.4.1 Gearbox Vendor Selection

Because of the limited space in the DRA shuttle, the rotational gearbox needed to be compact. The torque requirements on the rotation drive are very high, so the gear ratio needs to be large. Harmonic style gearboxes were chosen because of their compact design and available high gear ratios [29]. Harmonic Drive LLC was selected as the source for the gearbox for their wide selection.

4.4.2 Derivation of Mechanical Performance Equation

A geometric diagram of the rotating frame is sketched in Figure 4.8(a). In this diagram, the axes are attached to the center of rotation. The un-rotated center of the payload is at point **pl** at coordinates (X_{pl} ,Y_{pl}), and the un-rotated center of gravity is at point **cg** at coordinates (X_{cg} ,Y_{cg}). From the diagram, some convenient constants for points **pl** and **cg** are

Point **pl**:

$$\phi_{pl} = \tan^{-1}\left(\frac{Y_{pl}}{X_{pl}}\right)$$

$$r_{pl} = \sqrt{X_{pl}^2 + Y_{pl}^2}$$

Point **cg**:

$$\phi_{cg} = \tan^{-1}\left(\frac{Y_{cg}}{X_{cg}}\right)$$

$$r_{cg} = \sqrt{X_{cg}^2 + Y_{cg}^2} \, .$$

Newton's law in rotation form was be used to estimate the rotation time of the frame. A Free Body Diagram is shown in Figure 4.8(b), where θ(t) is the angle of rotation taken from the X axis to the top of the frame. Newton's law about the Z axis gives

$$\Sigma\tau_z = \tau_g - \left(\tau_{pl} + \tau_{cg}\right) = I\ddot{\theta}, \tag{4.23}$$

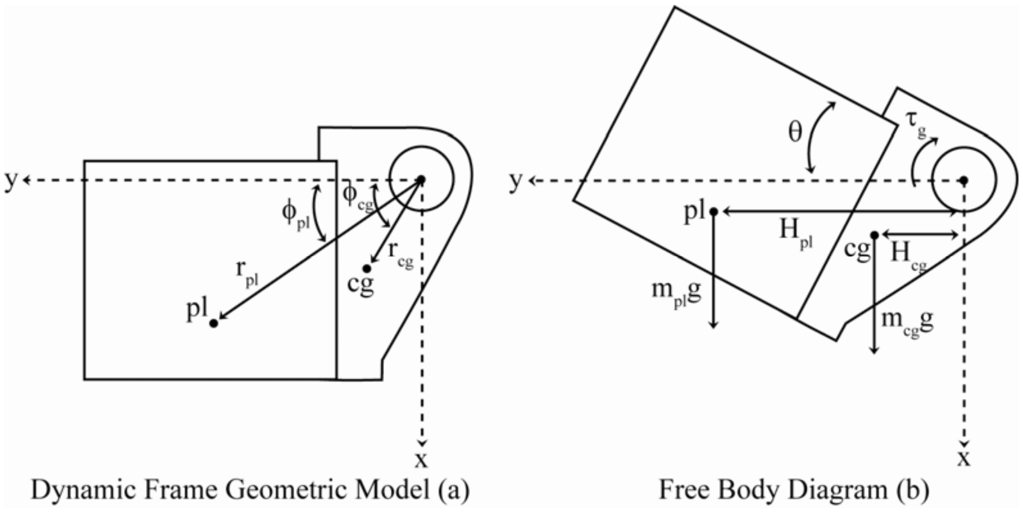


Figure 4.8: Geometric Diagram of Dump assembly

where τ_g is the torque applied by the gearbox, τ_{pl} and τ_{cg} are the torque about the Z axis from the weight of the payload and the dump assembly respectively. The moment of inertia *I* is the combined inertia about the Z axis for the shuttle (I_s) and the payload (I_p). Again friction has been neglected since a very high degree of accuracy is not necessary. The moment arms to points **pl** and **cg** are needed to calculate the torque τ_{pl} and τ_{cg}. Since gravity is always in the vertical direction, the moment arm is always the horizontal component of the location of the point. The moment arms are shown in Figure 4.8b as H_{pl} and H_{cg}. The moment arms and torques as a function of θ are

Point **pl**: Point **cg**:

$$H_{pl} = r_{pl}\cos(\theta - \phi_{pl}) \qquad H_{cg} = r_{cg}\cos(\theta - \phi_{cg})$$

$$\tau_{pl} = H_{pl} m_{pl} g \qquad \tau_{cg} = H_{cg} m_{cg} g\,. \tag{4.24}$$

All that is left to solve for in Equation (4.23) is the torque of the motor. This was shown to be equation (4.2) in Section 4.2.1, rewritten here

$$\tau_m = \tau_o - \frac{\tau_o}{\omega_o}\omega_m\,.$$

The output of this equation needs to be in terms of the torque after the gearbox and the input needs to be in terms of the gearbox speed in radians per second ($\dot{\theta}$). With the following substitutions, where N is the gearbox ratio,

$$\tau_g = N\tau_m \qquad \text{and} \qquad \dot{\theta} = \frac{\omega}{N}\,.$$

the torque output of the gearbox in terms of the gearbox speed becomes

$$\tau_g = N\left(\tau_o - \frac{N\tau_o}{\omega_o}\dot{\theta}\right). \tag{4.25}$$

With equations (4.24) and (4.25), equation (4.23) can be solved for $\ddot{\theta}$, and the mechanical system equation becomes

$$\ddot{\theta} = \frac{1}{I}\left[N\left(\tau_o - \frac{N\tau_o}{\omega_o}\dot{\theta}\right) - r_{pl}m_{pl}g\cos(\theta - \phi_{pl}) - r_{cg}m_{cg}g\cos(\theta - \phi_{cg})\right]. \tag{4.26}$$

This equation cannot be easily solved analytically because of the cos(θ) terms. This equation was solved numerically using Matlab and Simulink.

4.4.3 Electrical Performance Equations

The objective is to minimize the energy consumption of the rotation process. Following the same procedure for the vertical motor (section 4.2.2), the current and power are given by

$$i = i_{stall}\left(1 - \frac{K_e\omega_m}{V_a}\right) \tag{4.27}$$

$$E = \int Pdt = \int iV_a dt\ . \tag{4.28}$$

Since the torque requirement of the rotation motor is changing with the angle, the motor never reaches a steady state. Hence the current term in equation (4.28) cannot be pulled out of the integral, and the equation must be evaluated in its present form. This equation was solved numerically along with equation (4.26) by Simulink.

4.4.4 Overview of MATLAB and SIMULINK Code

Matlab and Simulink were use to solve equations (4.26) and (4.28) numerically [30]. Matlab code was used to establish constants, call the Simulink simulation, and plot the results. Simulink was used to integrate $\ddot{\theta}$ to get $\dot{\theta}$ and θ, which were fed back into the equation to get the next $\ddot{\theta}$. The Simulink simulation was called several times in the Matlab code: once for each payload paired with each available gear ratio. The simulation for a given trial terminated when the torque caused by the payload became zero, since by this point the payload is expected to be falling forward and simulating past this point becomes meaningless. The complete code is in Appendix A.

Values for Simulation

The rotating frame moment of inertia was calculated using the Pro/Engineer mass properties function [31], and the payload moment of inertia was approximated as a point mass at distance r_{pl}. The location for the center mass of the payload (X_{pl} ,Y_{pl}) was estimated from the Pro/E model based on where the payload might typically be centered and the location of the center mass of the rotating frame (X_{cg} ,Y_{cg}) was calculated using the Pro/E mass properties function.

Because of the selected manufacturer and series of gearboxes, the gear ratios are restricted to 50, 80, 100, 120, and 160:1. Each motor and gearbox combination was simulated for both the typical and maximum payloads.

Simulation Results and Generated Plots

The configurations which can rotate both loads are

- Motor 4100, gearbox ratios 120 and 160:1
- Motor 4106, gearbox ratios 160:1
- Motor 4300, gearbox ratios 100, 120, and 160:1
- Motor 4400, gearbox ratios 100, 120, and 160:1

The complete simulation results for these configurations are in Appendix A. Some typical results are shown here in Figure 4.9 and Figure 4.10.

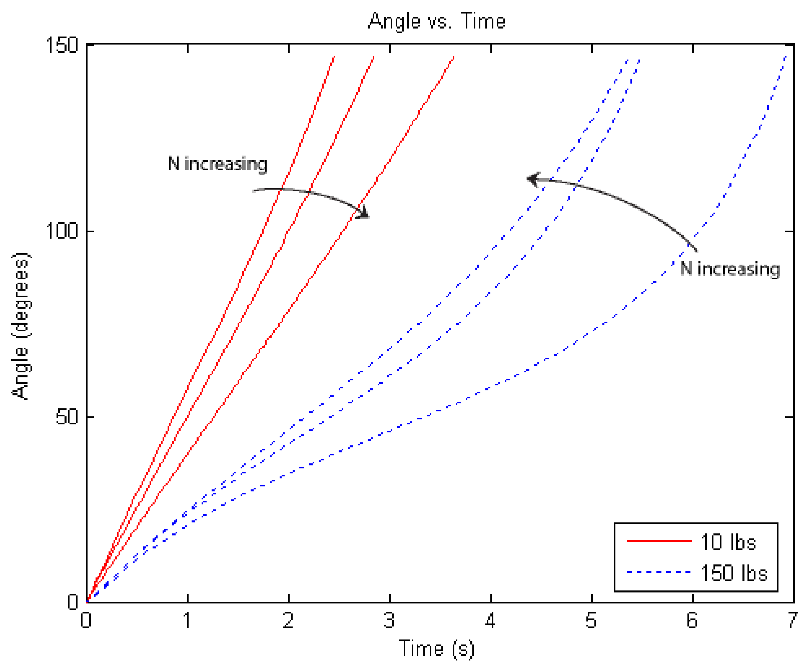


Figure 4.9: Angle vs. Time for Motor 4300, Gearbox Ratio 100, 120, and 160:1

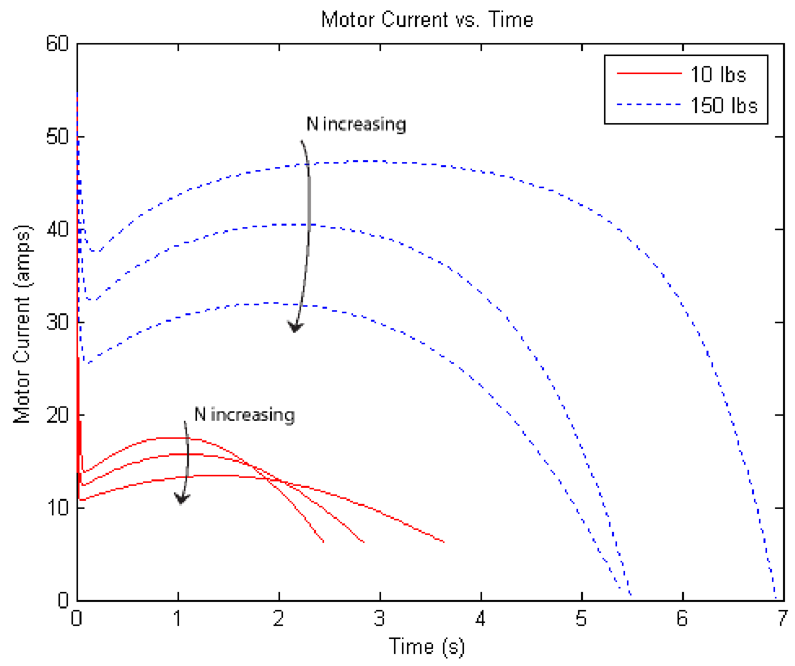


Figure 4.10: Current vs. Time for Motor 4300, Gearbox Ratio 100, 120, and 160:1

4.4.5 Narrowing of Rotation Motor

The constraints in Figure 4.2 were applied to narrow the available motors based on the simulation results in Appendix A.

Constraint 1: Dump time for the typical load must be greater than 2 seconds

Based on the generated plots, motor 4100 was eliminated for both available gearbox ratios.

Constraint 2: The dump time for the maximum load must be less than 7 seconds

Motor 4300 with gearbox ratio of 100:1 was eliminated.

Constraint 3: The continuous current must be kept below 60 amps

No motor/gearbox combinations were eliminated in this step, however, it is noted that motor 4100 would have been eliminated for both available gearbox ratios by this constraint if not by constraint 1.

Constraint 4: Must be able to brake maximum load at 0o

Given the dump frame model in Figure 4.8, the torque applied to the motor at 0^o with the maximum load is 283 Nm (2500 in-lbs). Motors 4300 and 4400 are 108XX series motors and have an available 3.95 Nm (35 in-lbs) brake. With this brake, all remaining gear ratios are able to brake the maximum load. Motor 4106 is an 80XX series motor which has an available brake of 15 in-lbs. With a gear ratio of 160:1, a braking force of 15.6 in-lbs is required. It was decided, however, that this was close enough since back-driving a 160:1 harmonic gearbox would present its own braking effect. In addition, the mechanical design of the DRA was modified to add a physical stop at 0^o and 180^o.

4.4.6 Final Selection of the Rotation Motor

This leaves the following available configurations

- Motor 4106, gearbox ratios 160:1
- Motor 4300, gearbox ratios 120 and 160:1
- Motor 4400, gearbox ratios 100, 120, and 160:1

Equations (4.27) and (4.28) were used to evaluate the energy used. The calculated energy used by motor 4400 with a gear ratio 100:1 and both motors 4300 and 4400 with a gear ratio 120:1 was larger than the same motors with a gear ratio 160:1 for both load cases. Thus all gear ratios except for 160:1 were eliminated based on minimizing the energy used. The energy used by the final 3 motors with 160:1 gear ratios is listed in Table 4-4.

Motor Number	Energy 6.5 kg	Used 68 kg
4106	89	1360
4300	110	810
4400	93	530

Table 4-4: Energy Used by Perspective Rotation Motors with 160:1 Gear Ratios

At this point in the analysis another constraint was added. It was determined that the 108XX series motors were too large to be incorporated into the DRA shuttle design without significant redesign. In the interest of keeping the development time down, these motors were removed from the analysis which included motors 4300 and 4400. The only motor that remained is 4106 with a gear ratio of 160:1 and is therefore the chosen rotation motor-gearbox configuration. The 15 in-lbs brake was selected with it.

4.4.7 Performance Calculations for Motor 4106

The simulated response of motor 4106 with a gearbox ratio of 160:1 is shown in Figure 4.11. The figure shows that the lift times are within the constraints. The current use is shown in Figure 4.12. The current use is well below the 60 amp limit.

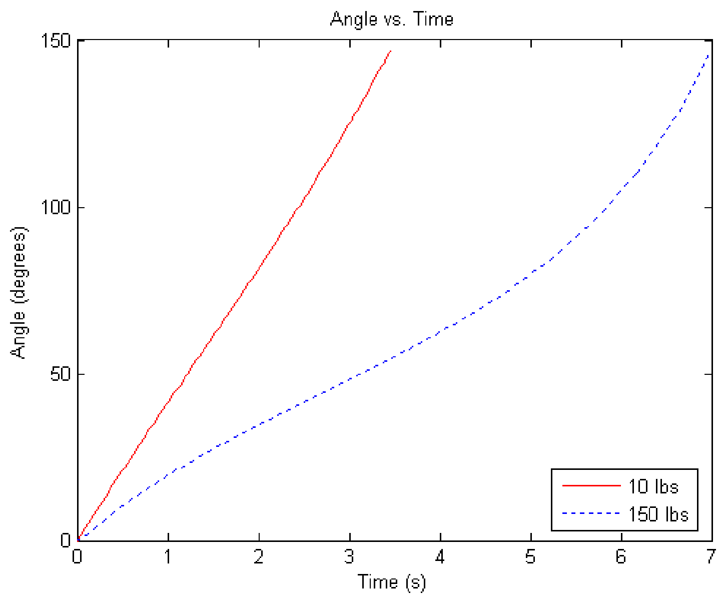


Figure 4.11: Angle vs. Time of 4106 with 160:1 Gear Ratio

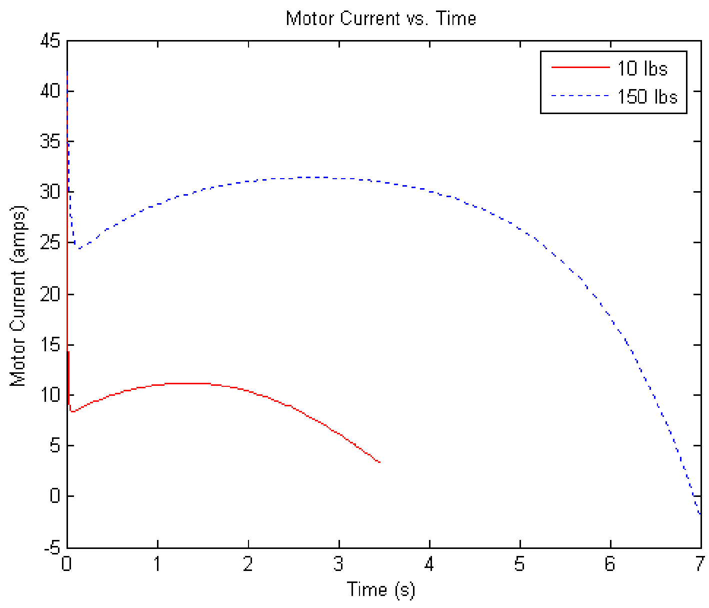


Figure 4.12: Current vs. Time for Motor 4106 with 160:1 Gear Ratio

4.5 Clam Gearmotor Selection

The clam motors were selected by a simpler method. The clams are not required to grip a litter bag in order to lift it, only to close the basket around the litter bag. For this reason, the clam motors had no real strength constraints because it is expected that the less powerful of Groschopp's motors will be sufficient for this. The only requirements was that the clams be able to close in a reasonable time and that the continuous current be below a specific threshold.

4.5.1 Mechanical Model for Clam Motor

The clams are not expected to be subjected to any significant loads, and because of the high gear ratio of the gearmotor, they get up to top speed very quickly and the motors can be assumed to reach no-load speed instantly. This assumption allows the 90° ($\pi/2$) closing time to calculated by

$$CloseTime = \frac{revolutions}{speed} = \frac{\pi/2}{\omega_o}. \tag{4.29}$$

The maximum closing force was calculated as the stall torque multiplied by the outer length of the clam (l), or

$$Force = \tau_o \cdot l. \tag{4.30}$$

)

4.5.2 Performance Calculations and Final Selection

Given that the clams are to close around a litter bag, it is expected that they are likely to be stalled if an obstruction prevents complete closure. In to minimize the risk to the microcontroller, which is joined to the clam motor controller, the Groschopp motor series with the smallest stall current was selected. This restricted the motor selection to 5 of Groschopp's low powered parallel shaft gearmotors. These also weigh less than the similarly powered planetary gearmotors, which will lighten the shuttle.

Equations (4.29) and (4.30) were used to calculate the data in Table 4-5. Based on the constraints in Figure 4.2, all but motor 50102 were eliminated.

Motor Number	Stall Torque (N-m)	No-Load Speed (RPM)	Close Time (s)	Single Clam Force (N)
50100	24.8	23	0.65	67
50101	32.8	16	0.94	88
50102	40.1	12	1.25	108
50103	87.0	6.2	2.42	234

Table 4-5: Calculated Clam Performance

With motor 50102, the maximum closing force from each clam is 108 N (24.3 lbs). The total closing force from both clams is then 216 N (48.5 lbs), which is sufficient for the requirements of the clam.

4.6 Summary

To select the vertical drive, the lift time needed to be estimated. Newton's Law, $F = ma$, was used to derive an analytical equation for position as a function of time which was simplified based on the assumption that the transient response of the motor is very quick compared to the entire lift time. This equation was used to enforce the time constraints on the motor selection. A function for the current used by the motor as a function of speed was obtained by assuming the inductive properties of the motor windings could be ignored. This was used to enforce the current constraints and also provided a means to evaluate the objective to minimize the energy. After applying the constraints and evaluating the objective, motor 63505 was selected, giving an estimated lift time for the typical and maximum loads of 6.7 and 10.1 seconds respectively. The motor was selected with a 1.69 Nm (15 in-lb) brake.

For the rotational motor, Newton's Law in rotational form, $\tau = I\alpha$, was used to derive a differential equation representing its motion. This equation was solved numerically using Matlab and Simulink. The same expressions were used to estimate the current and energy as for the vertical drive. Motor 4106 was selected with a harmonic drive gear ratio of 160:1, which gave an estimated rotation time for the typical and maximum loads of 3.4 and 7.0 seconds respectively. The rotation motor was also selected with a 1.69 Nm (15 in-lb) brake.

The clam motor was selected based on minimizing the current flow and keeping the closing time between 1 and 2 seconds. Motor 50102 was selected with an estimated closing time of 1.25 seconds and closing force of 108 N (24.27 lbs) per clam.

All the data sheets and CAD drawings for the chosen motors are in Appendix D.

CHAPTER 5: DRA CONTROL AND PROGRAMMING

5.1 Introduction

This chapter outlines the logic control and programming of the DRA's main controller, the OX2. The chapter begins by describing how the PS2 controller will be used to control the DRA and how to interface with the PS2 controller. Then the high level software architecture is presented and communication with the AX3500 is explained. The chapter ends with error detection and alerting protocols, including the use of vibration feedback.

5.2 Controlling the DRA with the PS2 Controller

The PS2 controller has two operating modes: analog and digital [16, 17]. In digital mode, the controller default, only the buttons can be read. To read the joysticks, the controller must be put into analog mode. This can either be done manually by the operator or automatically by the OX2. To make the DRA's operation as simple as possible, the OX2 was programmed to initialize the PS2 controller to analog mode and to lock it in analog mode. This ensures that the controller is in the correct mode and that the operator cannot unintentionally put it into digital mode, as this would produce undesirable results.

As stated earlier, the PS2 controller has two dual axis proportional joysticks. The vertical axis of the right joystick will be used to control the motion of the shuttle, and both the horizontal axes of the right and left joysticks will be used to control the clams. For the clams, moving the joysticks inward causes their respective clams to close, and moving the joysticks outward cause the clam to open. By using both joysticks, the clam motors can be operated independently of each other.

The vertical axis of the right joystick is used to actuate the shuttle. The right joystick vertical position will be interpreted by the controller into motor commands based on the shuttle orientation, as will be discussed in the next section. When the shuttle reaches the top of the track, the OX2 will automatically switch motors and begin rotating the shuttle without the operator changing their use of the PS2 controller. Likewise, when the shuttle has rotated 180^{o}, the OX2 will automatically switch back to the vertical motor and lower the shuttle without the operator changing their use of the PS2 controller.

Enable buttons must be pressed at the same time the joysticks are used. For the clams to be enable, the L1 button must be held, and for the vertical and dump frame to be enabled, the R1 button must be held. For the development phase, rotation of the dump frame can be executed by the operator independent of the top sensor configuration by holding L1, L2, R1, square, and X, and then moving the left stick vertically. Moving the joystick up rotates the dump frame in the positive direction (dump), and moving it down rotates the frame in the negative direction (deploy). The dump frame will still stop when the rotation limit sensors are tripped. This section of the code can be removed when no longer needed for the development phase.

5.3 Playstation 2 Controller Interface Overview

PlayStation 2 communication protocol is a bidirectional synchronous communication [32]. Command bytes are sent to the controller and data bytes are received back. Both command and data bytes are sent at the same time. The data communication is kept synchronous by a clock signal generated by the reading device (in this case, the OX2 microcontroller). There are a total of 9 lines in the PS2 controller cable; these are shown in Figure 5.1. (To avoid confusion, it is important to note that the PS2 controller cable in Figure 5.1 is different than the PS2 adapter in the wiring diagram of Figure 5-8, and each has a different wire color set that does not necessarily match). In order to communicate with the PS2 controller, a minimum of 4 data lines must be used: command (CMD), data (DATA), clock (CLK), and attention (ATT). Information is transmitted back and forth in 8 bit (1 byte) packages on the CMD and DATA lines. The CMD line is used to send information to the PS2 controller, and the DATA line is used to receive information from the PS2 controller. The ATT line is used by the reading device to select a specific controller if more than one is present. Even though there is only one controller used for the DRA, the ATT line cannot be ignored. The CLK line is used to keep the sending and receiving devices synchronized and must be generated by the OX2.

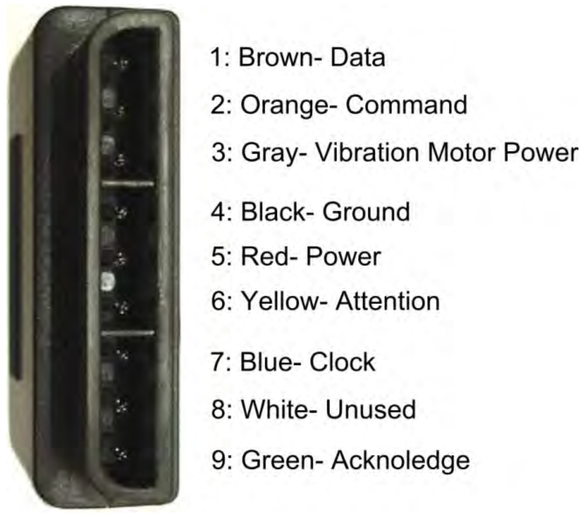


Figure 5.1: PlayStation 2 Wire Colors and Uses [32]

Information is sent and received through standard packets of between 5 and 21 bytes. Before a packet of information can be transferred, all communication lines must be set to logic high. To begin communication, the ATT line and CLK must first be pulled low by the microcontroller. All data and command bits are placed on their respective lines when CLK is pulled logic low, and all data and command bits are read when CLK is pulled high. All data and command bits are sent at the same time, so both devices are sending and receiving information at the same time. There should be an approximate 100μs delay between each level change of the CLK line.

At the end of transferring 8 bits, the controller pulls the acknowledge line low to signal to the reading device that communication was successful. This line also allows the reading device to determine if a PS2 controller is present and can be ignored. After a short delay of approximately 100μs, the next set of data and command bytes can be sent. When the entire packet of command and data bytes have been sent, the ATT line must be pulled high again to signal the end of the information transfer. To help illustrate PS2 controller communication protocol, a flowchart diagram is shown in Figure 5.2.

5.3.1 Data and Command Packet Protocols

There are specific set of commands that must be sent to initiate communication with the PS2 controller. Information transfer between the PS2 controller and the reading device always begins with a 3 byte header. In the header, the first command byte initiates data transfer and is always 0x01 (hexadecimal notation), the second command byte is the main command for the PS2 controller, and the third command byte is non-functional and is always 0x00. The second command byte tells the controller what the reading device wants from it and includes things such as: poll controller (0x42), enter/exit configuration mode (0x43), analog or digital mode select (0x44), and others of no use to this project.

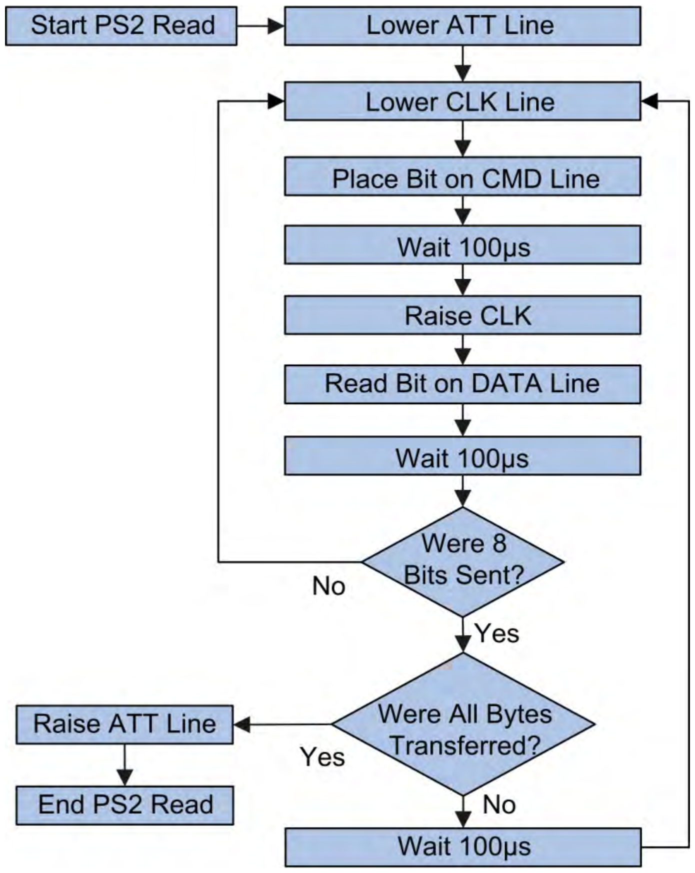


Figure 5.2: PlayStation Communication Protocol Flowchart

The standard header is documented in Table 5-1 for clarity, and the different main commands relevant for this project are listed in Table 5-2.

Byte #	Line	Discription
1	CMD	Always 0x01. Initiate information transfer
	DATA	Always 0xFF. Non-functional
2	CMD	Main command from reading device
	DATA	PS2 status ID byte
3	CMD	Always 0x00. Non-functional
	DATA	Always 0x5A. Non-functional

Table 5-1: Standard header for PS2 controller protocol [32]

The first returned data byte is always 0xFF and is non-functional, the second data byte is the PS2 status ID, and the third data byte is always 0x5A and is non-functional. The left hexadecimal digit of the PS2 status ID tells the reading device what mode the controller is in, including analog (7), digital (4), or configuration mode (F), and the right hexadecimal digit tells the reading device how many DATA words (2 bytes) will follow the header. So a PS2 Status ID of 73 means the controller is in analog mode and is sending 3 words of data after the header.

Main Commands	Function
0x42	Poll controller
0x43	Enter/Exit configuration mode
0x44	Analog or digital select mode

Table 5-2: Main Commands for PS2 Controller [32]

The header can be followed by 2 to 18 bytes, depending on the main command, subcommands, and the configuration of the controller. Subcommands modify the main command issued by the reading device. For example, the reading device can issue a main command of 0x43 (enter/exit configuration mode), and then issue a subcommand to enter or exit configuration mode (0x01 and 0x00 respectively). Once the controller is in configuration mode, a main command of 0x44 (analog or digital mode select) can be issued, then the subcommands 0x01 for analog mode and 0x03 to lock the controller can be sent. Typically, the data sent back by the controller while in configuration mode is 0x00 and can be ignored. The complete sequence of data and command bytes to put the PS2 controller in analog mode and lock it is shown in Table C.1 of Appendix C.

5.3.2 Vibration Feedback for Error Alert

The PS2 controller has a small vibration motor and a large vibration motor. The small vibration motor can only be turned on or off, while the large vibration motor can be varied proportionally [32]. This allows for various vibration strengths, though it is only necessary in this application to use full on or full off commands. Before commands can be sent to actuate the motors, the vibration function must be enabled while in configuration mode (main command 0x43). After the PS2 controller is in configuration mode, a main command of 0x4D must be sent to enter vibration function enable mode. After the header, the subcommands 0x00 and 0x01 are sent, which map the 0th and 1st bytes after the polling header to control the small and large motors respectively. After the PS2 controller has exited configuration mode, motor commands can be sent during the 0th and 1st bytes following the header of the polling command (main command 0x42).

The 0th command byte following the header turns on the small weight motor with a value of 0xFF (all others values issue a motor off command), and the first byte issues a proportional command to the larger weight motor with values 0x00 – 0xFF (full off to full on). The sequence to enable the vibration function is shown in Table C.2 of Appendix C.

5.3.3 Polling the Playstation 2 Controller

When the poll controller main command is sent, the first two data bytes following the header indicate which buttons are currently depressed. At the same time, the first two command bytes after the header issue commands to the vibration motors. Table 5-3 lists the button mapping for the two button configuration bytes, which are always bytes 4 and 5 of the controller poll. The bits returned by the PS2 controller are inverted, so a 1 represents an unpressed button, and a zero represents a pressed button. So if the controller returns a data byte 5 of 11011111 in binary, the L1 button is pressed and the clam motors would be enabled. When the controller is in analog mode, the last bytes 6 through 9 represent the right joystick vertical position, left joystick vertical position, right joystick horizontal position, and the left joystick horizontal position respectively. The values returned range in decimal from 0 for full left/down, to 255 full right/up. These values were modified to go from -255 to +255 for full left/down to full right/up. The complete sequence of data and command bytes to poll the controller are shown in Table C.3 of Appendix C.

Button	Select	L3	R3	Start	Up	Right	Down	Left
Byte.Bit	4.0	4.1	4.2	4.3	4.4	4.5	4.6	4.7
Button	L2	R2	L1	R1	Δ	○	X	□
Byte.Bit	5.0	5.1	5.2	5.3	5.4	5.5	5.6	5.7

Table 5-3: Button Mapping for DATA Bytes 4 and 5 [32]

5.4 High Level Control Software Overview

The control software for the OX2 was written entirely in the C programming language. A library of functions was written and grouped into header files corresponding to the purpose of the functions. In all, the header files for reading the PS2 controller, sending precise pulse commands to the AX3500, and general DRA functions like sending motor and brake commands were created. A header file for establishing RS232 communication was also created, but development of this was put on hold. The use of header files makes is easier to modify individual functions without modifying the main control software. There are also several standard header files available from Pololu such as SPI interface and delay functions which were also used [33]. The complete control software written for the OX2 can be found in Appendix B.

The main DRA code uses functions from the header files and established no functions of its own. It begins by initializing several system utilities such as the SPI interface, PS2 interface, and pulse interface. It also sets the internal pull-up resistors for the sensors and insures that the brakes are initialized to the locked position.

The main loop begins by reading the PS2 controller, which has previously been set and locked into analog mode by the initialization function. The OX2 checks to see that the controller is present and has successfully been put into analog mode by checking the return value of the PS2 status ID byte. If the ID byte is not the proper value, the OX2 assumes that the controller is not present or not in analog mode. The code then halts all motor operations and attempts to reestablish communication with the PS2 controller. This is to ensure that incorrect commands are not sent to the motors because of a not present or non-functioning controller. Checking of the PS2 controller ID happens every loop of the code so that if the controller connection is lost at any time, the OX2 can detect it and attempt to recover.

If the proper PS2 controller ID byte is returned, then the code proceeds to determine the configuration of the shuttle by reading all of the limit sensors except for the clam sensors. The clam orientation is ignored at this time since they move independently of the rest of the shuttle configurations. The code then executes a Switch flow control statement and branches to 1 of 8 functions based on the configuration of the DRA shuttle. The 8 shuttle configurations are: (1) shuttle at top of track and 0^o rotation, (2) shuttle at top of track and 180^o rotation, (3) shuttle at top of track and mid rotation, (4) shuttle at mid track and 0^o, (5) shuttle at mid track at 180^o, (6) shuttle at bottom, (7) shuttle docked, and (8) is an error configuration.

Configuration (8) will only be reached if there is an error in the sensor readings. If the error configuration is reached, the code exits the main loop and enters the error loop. This section of the code immediately halts all motor operations, which also enables the brakes. The code can never exit the error loop back to the main loop until the system is reset. This is discussed more in Section 5.6. The sensors to detect configurations (6) and (7) are not currently installed, but the software is setup to easily enable their respective portion of the code.

The rest of the switch statements begin by interpreting the PS2 controller input based on the configuration of the shuttle. The rotation and vertical motor speeds are then set accordingly. For example, if the sensors reported that the shuttle was at the top of the track and at 0^o rotation, the right joysticks vertical position will be interpreted as follows: if the right joystick has a positive vertical value (in the up position), a positive rotation speed and zero vertical speed is set, and if the right joystick has a negative vertical value (in the down position), a negative vertical speed and zero rotation speed is set. In this way the controller intelligently interprets the PS2 commands into motor commands based on the configuration of the shuttle. This all happens seamlessly without the need for the operator to change anything when the limits are reached. This insures that there is never a risk of interference do to improper operator handling.

The clam motor speeds are interpreted as the horizontal positions of the joysticks. The code is setup to allow the left and right joysticks to control the left and right clams or the right and left clams, respectively. This is needed due to the fact that when the operator is out of the truck, the right joystick needs to control the right clam motor, etc., but when the operator is inside the truck, the orientation needs to be reversed such that the right joystick controls the left clam motor etc. This is due to a change of perspective for the operator from inside to outside the truck. The operator can toggle which joystick controls which clam motor by pressing the top buttons L2 and R2, and the X button on the face of the controller simultaneously. This button configuration is complex enough to ensure that it should not be pressed by accident while still being easy enough to be executed by any operator. This control inversion can be performed by the operator at any time. The code is set up such that if the operator holds down the buttons, the inversion cannot toggle again until the buttons are released. This is needed since the main loop executes on the order of milliseconds, and if the buttons were held over more than one loop, the control inversion would continue to toggle.

Next, the controller checks to see which enable buttons are being pressed. If either or both buttons are not being pressed, the motor speeds associated with each enable button are zeroed. Hence if L1 is not pressed, the clam motor speeds are automatically zeroed, and if R1 is not pressed, the rotation and vertical motor speeds are zeroed regardless of the PS2 controller configuration. This ensures that no motion is allowed when the enable buttons are not pressed. Having separate enable buttons allows the clams to be actuated without unintentionally actuating the shuttle, and vice versa.

Last in the main loop, the code sends the motor commands to the two dual axis motor controllers, the AX3500 and the VNH2. If a nonzero speed is sent to the rotation or vertical motors, their respective brakes are automatically disabled by the OX2 controller. While testing the code, it was noticed that when the PS2 controller was close to sending a zero motor speed, the OX2 would chatter the brakes on and off. To avoid this, a section of code was added that monitored if the brake configuration had just been switched, and if so, a delay of 50ms was carried out. This helped to stop the chatter. The DRA control software architecture is summarized in a block diagram in Figure 5.3.

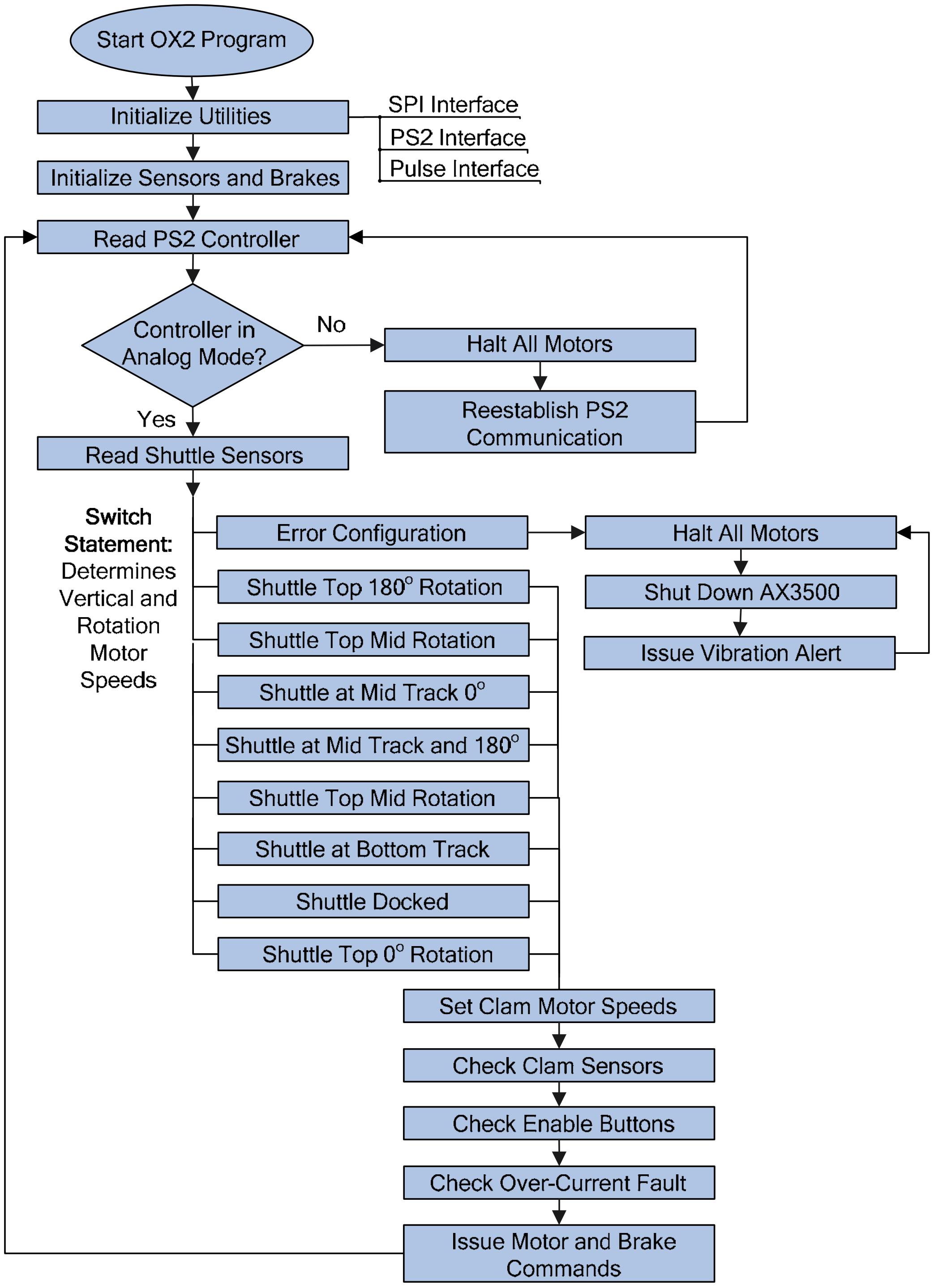


Figure 5.3: DRA control software architecture overview

5.5 Communicating with the AX3500

The AX3500 has three communication options: (1) 0-5 volt analog control, (2) RC pulse protocol control, and (3) two way RS232 serial control [13]. RS232 was the communication method of choice for reasons stated in Section 3.4.1; however, difficulties were encountered when programming the OX2 for serial communication. In the interest of time it was decided to move ahead with the project using RC pulses and return later to develop the RS232 communication. Both methods of communication are presented here.

5.5.1 RC Pulse Interface Overview

RC pulse protocol was developed for the remote control (RC) vehicle hobbyist community to communicate between the RC receiver unit and electric motors and servo motors [13]. The command is determined by the duration of a pulse on the motor command line. A 1.5ms pulse represents a stop command, 1.0ms represents a full reverse command, and 2.0ms represents a full forward command. The values between allow for proportional control. The receiver expects a pulse every 20ms, though it is not critical and most receivers will accept pulses much further apart. This protocol is illustrated in Figure 5.4.

This pulse pattern was created in the code by simply pulling the proper pin high, then delaying the code for the desired pulse duration, and pulling the pin low again. The entire main loop executes in faster than 20ms, so a delay was inserted at the end of the code to ensure a pulse was sent approximately every 20ms. An oscilloscope was used to fine tune the duration and frequency of the pulses.

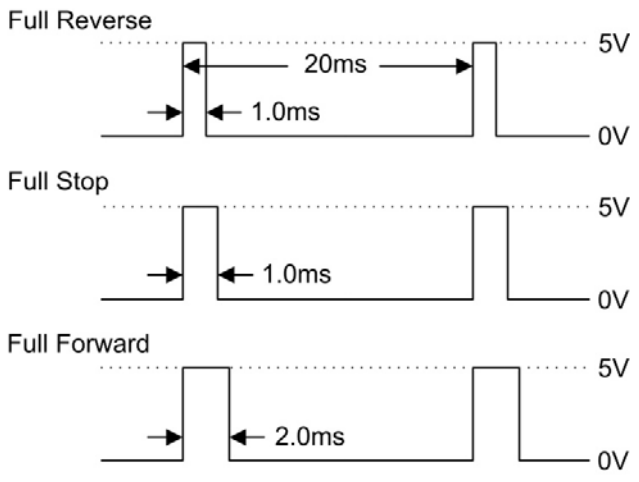


Figure 5.4: RC pulse communication protocol

5.5.2 Serial RS232 Interface Overview

RS232 is a bidirectional serial protocol commonly used in computer systems [34]. It is a -12V to +12V logic level signal, with -12V representing a logical 1 and +12V representing a logical 0. In order for the OX2, which is a 0-5V TTL device, to communicate at these voltage levels, an adapter will be used. The adapter was acquired from Pololu and converts the TTL signals to the inverted -12V and +12V signals. With this adapter, the OX2 can communicate with the AX3500, which requires the -12V and +12V logic level signals.

To produce the serial data bits, the ATmega644 microprocessor of the OX2 unit has on-chip hardware called a USART (universal synchronous/asynchronous receiver/transmitter) [20]. This hardware allows the OX2 to generate a wide range of synchronous and asynchronous serial communication protocols. To use the USART, several special purpose memory registers of the ATmegga644 must have the appropriate values written to them. The UBRR0H and UBRR0L registers must be set to the appropriate values in order to produce the desired baud rate (bits/second), which for RS232 is 9600 baud. The values of UBRR0H and UBRR0L are dependent on the desired baud rate and the clock frequency of the microcontroller. The appropriate values can be found in the ATmegga644 data sheet for a wide range of clock frequencies. The baud rate is critical for asynchronous communication since there is no clock signal to synchronize the two devices. The ATmega644 must also be set to send an even parity bit and 7 data bits, which is set in the UCSROC special purpose register. The ATmega644 defaults to 1 start and 1 stop bit, which is required for RS232 serial protocol. Lastly, the receive and transmit data lines must be activated it the UCSROB special purpose register. The receive and transmit lines are pins D0 and D1 on the OX2 respectively.

5.5.3 Sending and Receiving Commands

The AX3500 commands and queries are made up by sets of 2 to 4 characters followed by the shuttle return value [13]. Commands begin with the "!" character and queries begin with the "?" character. A subset of the commands and queries relevant to this project are listed in Table 5-4. In the table, "nn" and "mm" represent numbers in hexadecimal. The AX3500 will echo back every character sent to it to acknowledge a successful communication. After the shuttle return character, if the command was received successfully and was valid, the AX3500 sends back a "+" character. If the transmission was unsuccessful or invalid, the AX3500 sends back a "-"

character. Looking at Table 5-4, if the command !AFF was sent, channel 1 (vertical motor) would be full forward and the character "+" would be returned by the AX3500.

Command/ Query	Discription	Returns	Note
!Ann	Channel 1, forward command	+	nn is in hexidecimal
!ann	Channel 1, reverse command	+	nn is in hexidecimal
!Bnn	Channel 2, forward command	+	nn is in hexidecimal
!bnn	Channel 2, reverse command	+	nn is in hexidecimal

?A	Read motor amps	nn
		mm
			nn = motor 1 amps in hex
			mm = motor 2 amps in hex

?M	Read heatsink temperature	nn
		mm
			nn = MOSFET 1 temp in hex
			mm = MOSFET 2 temp in hex

Table 5-4: AX3500 command and query subset [13]

5.6 Error Protocols

If an error is encountered, the OX2 must detect it and respond appropriately. There are 4 errors that the OX2 will be able to detect: (1) an invalid shuttle configuration, (2) the PS2 controller not present or nonfunctional, (3) an over-current fault for either of the clam motors, and (4) an over-current fault for the rotation or vertical motors. The last error will not be detectable until the bidirectional RS232 communication is established (see recommendation in Section 7.3.4).

The code is also not presently set up to respond to the current levels in the clam motors, though this is easy to implement later. In the case of an over-current fault in any of the motors, the OX2 should halt the motor and issue a vibration alert to the operator.

If the OX2 reads an invalid shuttle configuration, the program exits the main loop and enters the error loop, which will halt all motor operations and issue a vibration alert to the operator. Reasons for a faulty shuttle configuration may include (1) a broken sensor, (2) a physical obstruction, (3) damage to the DRA, or (4) an improperly aligned sensor. Even if the sensor readings become valid again, the code can never return to the main loop without a power reset of the OX2. This is to ensure that if a faulty sensor configuration is detected, that the operator must actively reset the system. This will encourage the operator to search out for the cause of the sensor fault. The code does not yet implement the other error loop action shown in Figure 5.3 of disabling the AX3500. This can be integrated later and requires an external logic power supply for the AX3500 which can be switched on and off by the OX2 (see recommendation in Section 7.3.1).

CHAPTER 6: DRA CONSTRUCTION, WIRING, AND TESTING

6.1 Introduction

This chapter discusses the assembly, wiring, and testing of the DRA. The chapter begins with a very brief description of the DRA's construction and assembly, as well as some assembly issues and their resolution. Next, the wiring of the DRA unit and electronics enclosure is discussed, and software debugging and verification is also presented. Last, test bed testing of the DRA is discussed and the measured performance data is compared to the calculated performance data of Chapter 4.

6.2 DRA Construction and Assembly

The parts for the DRA were constructed by out-of-house machine shops. All of the large tolerance parts were made by water jet machining and the small tolerance parts were made by traditional machining methods. The weldments were also welded out-of-house, and then the parts and weldment subassemblies were assembled in the AHMCT robot lab. The unit was mounted on a test bed for final assembly, wiring, and testing. The complete DRA assembly on the test bed is shown in Figure 6.1 left, with a close up of the dynamic assembly in Figure 6.1 right. Figure 6.2 left shows the DRA with the shuttle midway up holding a litter bag, and Figure 6.2 right shows the DRA with the shuttle down and gripping a couple of boxes.

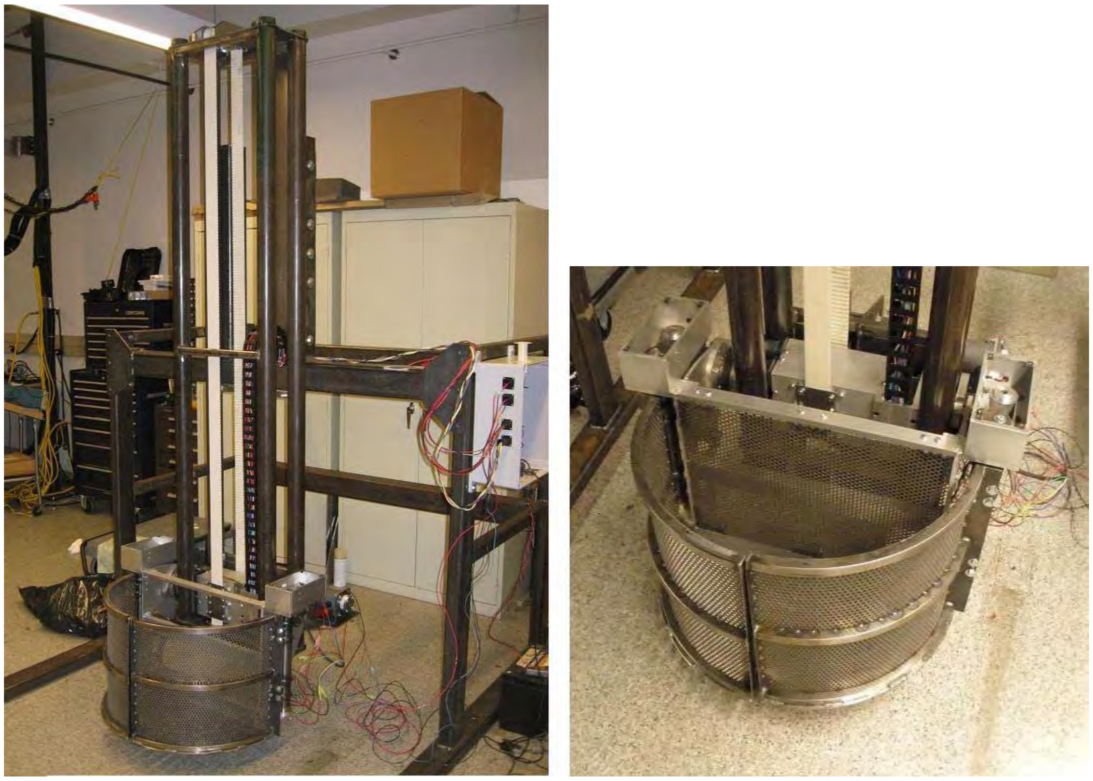


Figure 6.1: DRA Assembly and Test Bed (Left) and Dynamic Assembly (Right)

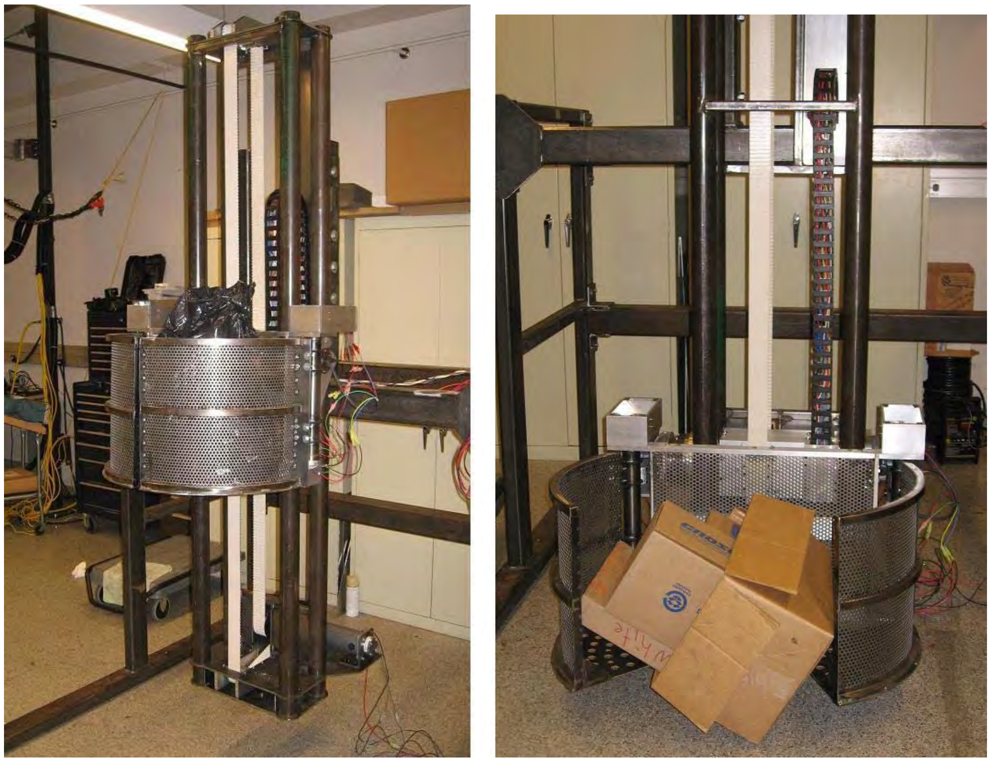


Figure 6.2: DRA Assembly Midway With Bag (Left), Lowered With Box (Right)

6.2.1 Assembly Issues

There were several issues encountered while assembling the DRA. Several parts needed to be modified in order to be fit properly. Particularly, the main timing belt assembly had a couple of issues. The lower idle roller shaft closest to the vertical motor would not fit into the bearings and needed to be turned down a fraction of a millimeter (few thousandths of an inch). The vertical motor, which is at the base of the DRA unit, is designed to slide a several centimeters horizontally in order to tension the belt. Even at the lowest tension setting, the last idle pulley could not be assembled with the belt on because the belt was too tight. To allow for assembly, the upper idle pulley mounts were shortened 9.5 mm (3/8 inch) to shorten the total belt travel length to reduce the tension on the belt. After this, the pulleys could all be assembled and the belt was tensioned by pulling the vertical motor back and tightening it.

The left and right clam boxes, which house smaller timing belts for clam power transmission, also needed some modification. The holes in the boxes for the clam shafts needed to be opened up in order to allow the shaft bushings to fit properly. The inner diameter of the shaft bushings also needed to be opened up for the clam shafts to fit. There were also some issues with friction when the boxes were bolted to the shuttle assembly. When the clam boxes were bolted down, they pressed onto the clam bearing causing some radial force on the clam shafts. This friction greatly reduced the clam motor closing force and needed to be reduced. The bushings were shortened a fraction of a millimeter such that they no longer contacted the clam boxes when the boxes were bolted down. This greatly reduced the left and right clam friction to reasonable levels.

There were issues assembling the right shuttle rollers which ride between the vertical tubes. It turned out that the outer diameter of the track tubes were slightly too large (on the order of hundredths of millimeters), which was just enough to prevent all of the rollers from assembling easily. The rollers could be assembled with difficulty, but the friction due to the improper fit was significant. The first thought to address this was to sand down the rollers, but this was not ideal since the rollers had a specific profile to allow them to roll on the track tubes. It was eventually decided to pry at the roller assembly until the rollers could all be put on to let the rubbing of the rollers on the tubes wear them down slowly.

6.3 Wiring the DRA

The DRA was wired with discrete wires available in the AHMCT facilities. (See recommendation in Section 7.3.2 for wiring improvement.) The motors were each wired with red and black 14 gauge multi-core wire. The sensors and brakes were wired with different color smaller gauge wires. Except for the internal to external electronics enclosure interface, all of the wire-to-wire connections were made with 0.25" quick disconnects (see Figure 6.3) [35]. This enables easy and nondestructive wire separation for any assembly or maintenance procedures.

The left clam sensor and left clam motor wires were passed inside the square stock tubing on the dump assembly to the right side where they could be connected to the rotating side of the slip ring. Also connected to the rotating side of the slip ring are the right clam sensor and right clam motor. All of the wires off of the stationary side of the slip ring, the rotation motor wires, the rotation brake wires, and the wires for the rotation sensors and top sensor were connected to discrete wires which were passed through cable carrier and then through the plate on the back of the DRA. The wires for the vertical motor and vertical brake were all passed along the back of the DRA and collected with the shuttle wires exiting the cable carrier. All of the wires were collected and routed directly to the electronics enclosure.

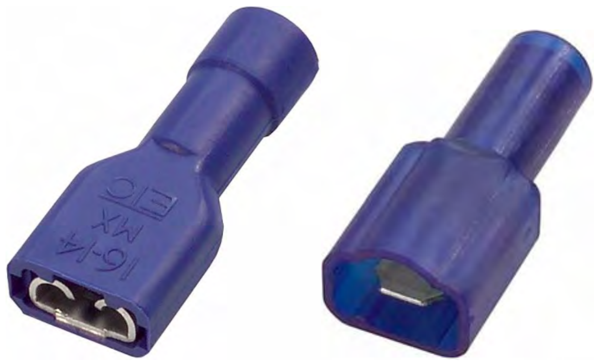


Figure 6.3: Male and Female Wire-to-Wire Quick Disconnects [35]

6.3.1 Wiring the NEMA Box

A NEMA (National Electrical Manufacturers Association) 4X rated enclosure was selected to house all of the electronics [36]. NEMA 4X enclosures are for indoor and outdoor use and are rated to protect against splashing, washdowns, dust, and corrosive agents. All of these conditions are expected to exist in the environment of a work truck to which the enclosure will eventually be attached. A fiberglass enclosure was selected so that it would not interfere with any wireless communication within the box. The NEMA box selected is shown in Figure 6.4 and has external dimensions of 386mm x 335mm x 180mm (15.2”x 13.2”x 7.1”), has a lift-off cover and mounting plate dimensions of 328mm x 277 (12.9”x 10.9”) [37].

One last internal component added was a standard size ATC (blade style) fuse block [38]. This was not mentioned in Chapter 3 because it is a passive component. A simple 8 circuit ATC style fuse block by Buss, model 15600-08-011, was selected and was mounted to the enclosure plate (see Figure 6.5 for a 10 circuit version). A different circuit was used for each electrical component: 3 amps for 5V regulator, 3 amps for brake relay (the brakes were measured to draw 800 milliamps each), 15 amps for the OX2, and 60 amps for the AX3500. Since each circuit of the fuse block is rated at 30 amps, two 30 amp fuses were used in parallel for the AX3500.

AMP Series 1 and 2 connectors by Tyco Electronics were used for the NEMA enclosure [39]. The connectors allow for a quick disconnect of the wires from external of the NEMA box. An example connector is shown in Figure 6.6.

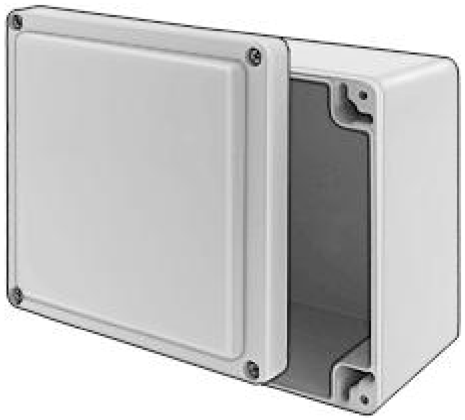


Figure 6.4: NEMA 4X enclosure [37]

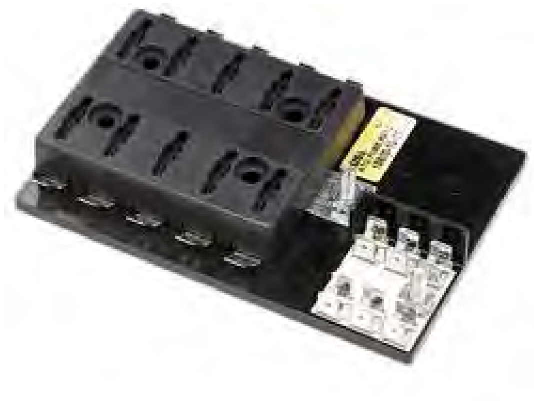


Figure 6.5: ATC Fuse block by Buss [38]

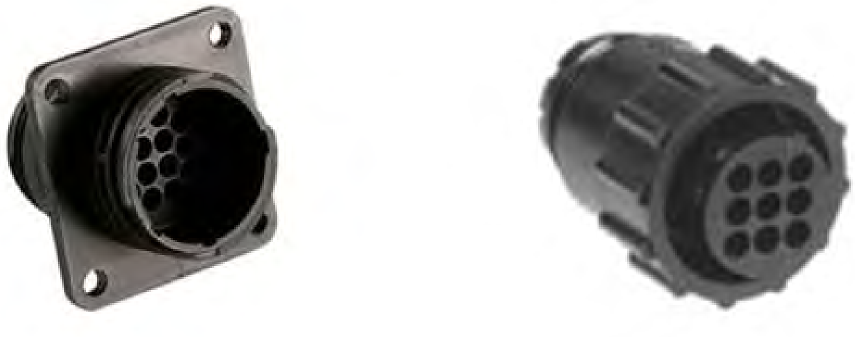


Figure 6.6 AMP Series 2 male and female connectors [39]

Pro/Engineer was used to plan the electronics layout inside the enclosure [31]. A part was created of the appropriate dimensions for each electronic component and placed together in an assembly. The NEMA box was also modeled to help with the amp connector placement. From there, wires were created and routed using a cable application within Pro/Engineer. After the model was finished, the components and connectors were adjusted until a sufficiently clean layout was obtained. The assembly model with the wire layout is shown in Figure 6.7 (left). This was mainly used to aid in the layout of the electronics and was not used to determine the exact path of the wires. The fully wired enclosure (except for the PS2 cable connector) is shown in Figure 6.7 (right). For the complete DRA wiring diagram, see Figure 6.8.

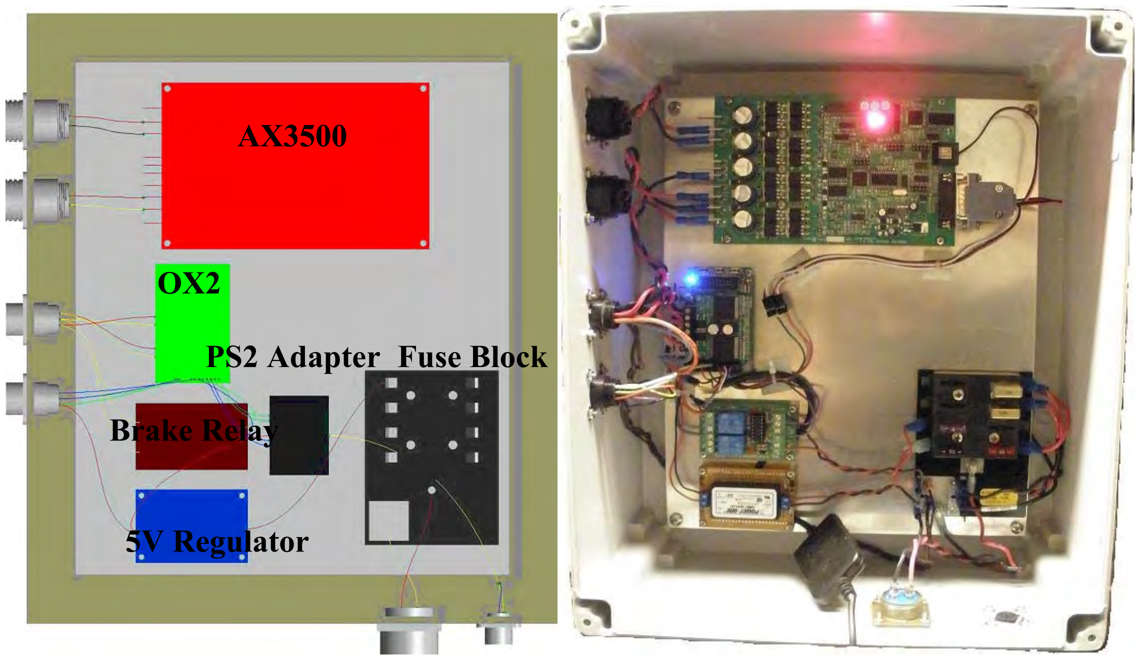


Figure 6.7: NEMA Box Pro/Engineer Model (left), Actual NEMA Box wiring (right)

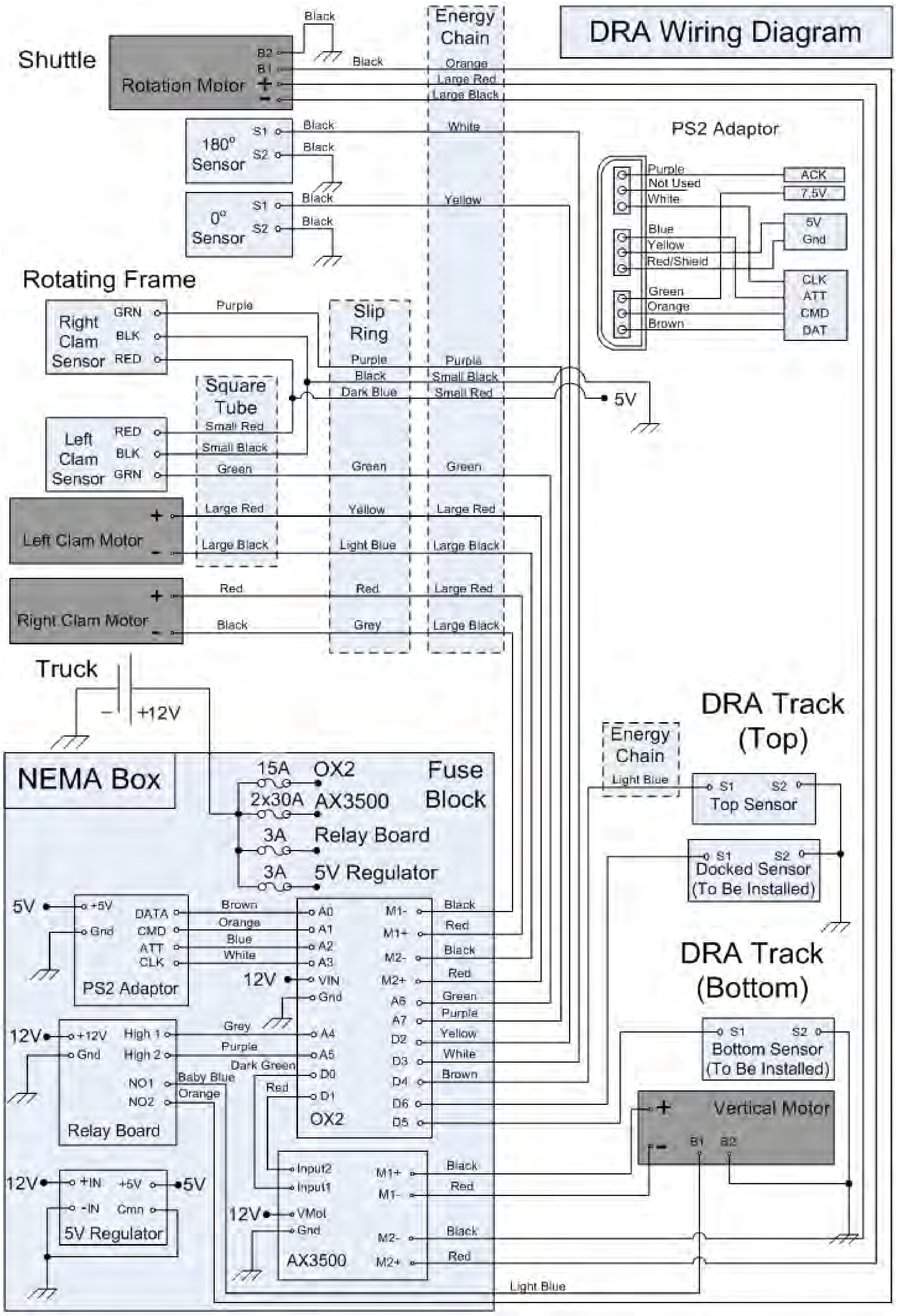


Figure 6.8 DRA wiring diagram

6.4 Software Testing and Corrections

Final software testing began after assembly and wiring of the DRA was finished, where a few errors were found. When the OX2 was first powered up, the vertical brake would switch off for a fraction of a second. This could cause the shuttle to move down slightly when first powered on. Initially the cause could not be located, as the brake initialization section at the beginning of the code was verified to be correct. Another strange behavior was found, which was eventually linked to the same software bug as the startup brake flicker. When checking the OX2's response to a disconnected PS2 controller, the brake switched on and off rapidly until the controller was reconnected. The OX2 was able to reestablish communication with the PS2 controller as intended, but the brake flickering behavior was obviously an unacceptable behavior. Knowing that the brake flicker was associated with a disconnected PS2 controller, the error was traced back to the PS2 initialization code.

Originally, the ACK line was to be used when communicating with the PS2 controller, but was later discarded when it was determined unnecessary. The PS2 communication code, however, still established an IO pin for the ACK line and initialized it to a logical high. The IO pin that was originally to be used for the ACK line is now used for the vertical brake IO pin. This meant that every time the PS2 initialization code was called, the ACK pin, and therefore the brake pin, was pulled high. When this piece of code initializing the ACK line high was removed, both brake switching issues were eliminated.

An important change in the code was made to help make DRA operation more robust. In several parts of the code, the OX2 looks for specific buttons presses, such as the clam enable button L1, the shuttle enable button R1, and the clam control switch buttons L2, R2, and X. The code originally ignored other buttons such that if L1 was being pressed with other buttons, the clam motors would still be enabled. It was decided that DRA operation would be more robust and tolerant of accidental button presses if the code would only accept exact button presses. Thus, if the buttons L1 and L2 were accidentally being pressed, the clam motors would not be enabled; L1 must be pressed by itself to enable the clam motors. This helps prevent accidental DRA motion due to random button pressing, which is likely to occur during normal operation.

Other minor errors were also encountered and corrected such as incorrect motor directions and reversed brake controls.

6.5 Initial Lab Testing

Subsystems were first tested individually to ensure each functioned as intended and to look for any faults or potential issues. First the clam subsystem was verified. The clams were fully opened to ensure that the OX2 stopped the clams from opening further once the ferrous vane sensor had been tripped. This was important since if the OX2 allowed the clams to continue undesired interferences would occur. If needed, the tripping point of the clam sensors can be adjusted to compensate for a vehicle body once the DRA unit is placed on a truck.

The system was then put through complete unloaded operation cycles to see the DRA's operation and to check for any mechanical or design faults. Initially there was some concern as to how smooth operation would be since the brakes are enabled immediately after a zero speed is issued. It was thought that the starting and stopping motions would be very rough and abrupt. Initial trials showed that the vertical operation was moderately smooth at the start and stops, even with abrupt brake enabling. The friction is inconsistent along the track, which causes the shuttle speed to be somewhat variable. This is discussed in more detail in the next section.

The shuttle rotation motion was found to be very smooth and consistent. The clearance was estimated to be about 6 mm (1/4") at the closest points during rotation. This requires the shuttle be at the absolute top of the track before rotation begins, which was important for the top track sensor alignment. Both brakes were found to hold very well with no slipping. Overall, initial lab testing showed the operation of the DRA to be moderately smooth and robust.

6.5.1 Operational Issues

It was found that the shuttle belt drive system was tracking away from the vertical motor. When the shuttle was driven up and down the vertical track a few times, the end flange of the vertical motor pulley came off because of the axial force caused by the tracking belt. It is believed that the tracking was due to two reasons: (1) there is some play in the orientation of the vertical motor due to the belt tensioning mechanism, the vertical motor axis was not perfectly perpendicular to the belt subsystem, and (2) the idle roller closest to the motor has no end flanges, which allows the belt to slide off. Once the belt begins to slide off even a small amount, the tension in the belt causes it to continue sliding off. To fix this, the motor was repositioned to be more perpendicular to the belt to reduce tracking. Flanges were also added to the idle pulley, which will be constrained from sliding itself by spacers. These combined changes eliminated the belt tracking issue.

Another issue experienced was an unexplained lack of power in the clams. The force generated, ignoring friction, was calculated to be 108 N (24.3 lbs) for each clam at the outermost point of the clam. Yet the force produced in the actual system was estimated to be on the order of tens of newtons (couple of pounds), with the left clam producing virtually no force. Friction could have been one issue reducing the clam systems power, but the clam system friction was tested by feel and was not appreciable to account for the force loss. To test the friction, the motor was disengaged from the clam system, while everything else remained in place, and the force to move the outermost point of the clam was negligible. Another reason it could not be due to friction was that the unloaded clams were still moving at an appropriate speed as if they were not loaded at all.

When stalled, the motors sounded as if the internal motor was spinning independent of the locked gearbox output. This gave reason to believe that the motors may not have been coupled properly with the gearbox. Another observation was that when both motors were stalled, the clam motors were not using the expected amount of current, which should have been the stall current 7.3 amps. They were measured to be pulling between 2 and 3 amps each. The motors were sent back to the manufacturer where some repairs were made.

It was assumed that the gearboxes were defective or insufficiently coupled to the motor shaft, and that this was the cause of the underpowered clam motors. Careful consideration lead to the possibility that the motors are being used in a manner which they were not intended for: the motors were often being back-driven as the clams were open and closed manually, the clam motors were often stalled when closed or when measuring their force, and the motors are subjected to a radial load on the output shaft. It is possible that through these actions, the gearboxes are being decoupled from the motor shafts. It will be important to observe if the clam motor performance changes with time. It may be necessary to replace these motors with others that are better suited to operate under these conditions.

An issue that made assembly difficult reappeared in the early testing phase. Binding and friction effects were encountered at the very bottom and top of track, which increased the force it took to raise the shuttle. One of the main causes of this was that the shuttle rollers were being forced to the side by an axial force and were rubbing against the shuttle body. It was decided that spacers would be added between the rollers and the aluminum sides to prevent rubbing, though this had not been implemented at the time of finishing this report. As a temporary fix to reduce friction, WD-40 was sprayed between the wheels and the body. This helped reduce friction so that performance values could be taken.

6.5.2 DRA Performance Data

The force generated by the clam motors was first measured. The force generated by the clam motor at the outermost point of the clam basket was measured perpendicular to the rotational axis. The force was measured somewhat crudely with an industrial electric scale (Pelouze model 4040) [40] by manually holding the scale up to the appropriate section of the clam and maxing out the closing force of the motor. The force measured was found to be highly variable and inconsistent, varying from a high of 102 N (23 lbs) to a low of 67 N (15 lbs), generally being around 85 N (19 lbs). This variability must be caused by the motor, the motor controller, or the measurement method. To rule out the measurement method, which seems the most likely, a better force measurement approach should be used. The 90^{o} closing times for the left and right clams were measured to be 1.35 and 1.38 seconds respectively. The measured data for the clam system is shown in Table 6-1.

	Left Clam	Right Clam
Max Measured Force	102 N (23.0 lbs)	93 N (21.0 lbs)
Calculated Force	108 N (24.3 lbs)	108 N (24.3 lbs)
Measured Close Time	1.35 seconds	1.38 seconds
Calculated Close Time	1.25 seconds	1.25 seconds

Table 6-1: Clam Performance Data

Lift and rotation cycle times for the DRA were also measured. (Note that the calculations for the vertical lift time of Chapter 4 were based on 1.83 m (6 ft) of travel, while the actual manufactured track has 1.88 m (6 ft 2 in) of travel.) It was not deemed safe to test the full capacity of 68 kg (150 lbs) in the lab, so only unloaded, 4.5 kg (10 lbs), and 13.6 kg (30 lbs) trials were carried out. The weights used were standard disk weights. Five sample times were taken for each trail to get an average and standard deviation. The average vertical lift trial times for unloaded, 10 lbs, and 30 lbs were measured to be 9.48, 9.94, 11.36 seconds respectively.

The rotation times were also measured for various loads. The weights were bolted to the bottom of the clams so that they could be rotated without slipping out. To simplify the measurement taking process, the time for the shuttle to rotate from 0^{o} to 90^{o} was taken for unloaded, 4.5 kg (10 lbs), and 13.6 kg (30 lbs) cases. The average rotation times for the three cases were 2.13, 2.28, and 2.59 seconds respectively. The rotation simulation of Chapter 4 was re-executed with a payload of 13.6 kg (30 lbs) to have a direct calculation comparison to the third rotation trial here. The 5 trial times for the vertical and rotation axes, their average, and standard deviation are in Table 6-2. The average values of the measure data is shown in Table 6-3 alongside the calculated times and one applicable design goal (all calculated and design goals for the vertical times are adjusted to compensate for the actual track length).

		Unloaded	4.5 kg (10 lbs)	13.6 kg (30 lbs)
Vertical	Trial 1	9.57	9.95	11.45
	Trial 2	9.45	9.91	11.35
	Trial 3	9.57	9.92	11.29
	Trial 4	9.51	9.96	11.45
	Trial 5	9.29	9.97	11.24
	Average	9.48	9.94	11.36
	St.Dev.	0.12	0.03	0.09
Rotational	Trial 1	2.16	2.23	2.65
	Trial 2	2.09	2.16	2.65
	Trial 3	2.09	2.34	2.44
	Trial 4	2.15	2.37	2.64
	Trial 5	2.14	2.29	2.58
	Average	2.13	2.28	2.59
	St.Dev.	0.03	0.08	0.09

Table 6-2: Vertical and Rotation Measured Lift Times

	Vertical			Rotation	
Cases	Measured	Calculated	Design Max	Measured	Calculated
Unloaded	9.48	6.68	N/A	2.13	2.14
4.5 kg (10 lbs)	9.94	6.89	7.19	2.28	2.23
13.6 kg (30 lbs)	11.36	7.19	N/A	2.59	2.43

Table 6-3: Vertical and Rotational Performance Data and Calculations

6.6 Comparison of DRA Performance to Calculations

The clam forces were a little less than the calculated value. This is easily accounted for by the moderate friction in the clam shafts. As noted earlier, it may be possible that the motors are not being used in a manner they are intended for, and their performance over time should be observed.

The vertical times are a slower than the calculated times, with the discrepancy increasing quite a bit from the 4.5 kg (10 lbs) to 13.6 kg (30 lbs) trial. The discrepancy can be accounted for by the friction and binding in the u-groove wheel system, which was ignored in the calculations. As mentioned in the section on operation issues (Section 6.5.1), binding and friction issues were encountered at the top and bottom of the tubes. The shuttle vertical velocity across these areas was noticeably slowed due to the friction. As progress is made to minimize and correct these issues, the travel times should continue to improve, though the travel times will always be less than the ideal calculations of Chapter 4.

The rotational times measured were more comparable to the calculations from the simulations in Chapter 4. This can be attributed to the low friction and high torque harmonic gearbox and its direct coupling to the dump frame.

6.7 Simulated Environment Testing

To test the viability of the system, an environment that simulated the actual picking up of bagged debris was created with a repeatable testing plan. This let us test multiple iterations of the DRA's design with multiple types of bags to simulate most of the extreme conditions that it would encounter in typical daily usage. This section will go into detail on the test plan implemented and the design process of successfully implementing the DRA in a "real-life" environment.

6.7.1 Overall Test Plan Implementation

The initial test plan went as follows: set up the DRA in a controlled stationary setting and evaluate design efficacy using bagged debris, iterate on design and retest if needed, and attach to vehicle and try a preliminary controlled road test. For each of these sections of testing, four "types" of bagged debris were tested that were felt to best simulate real life. These bags included: a bag that was heavy and had a small amount of trash in it (filled with items such as phonebooks) weighing 10.5 pounds, a bag that was light and had a small amount of trash in it (filled with items such as fast-food cups, wrappers, and balled paper) weighing 2.5 pounds, a bag that was heavy and had a large amount of trash in it (filled with items such as phonebooks, fast-food cups, wrappers, and balled paper) weighing 9 pounds, and a bag that was light and had a large amount of trash in it (filled with items such as dead vegetation, fast-food cups, wrappers, and balled paper) weighing 3 pounds. It is of note that all bags tested were heavy-duty orange Caltrans "Don't Trash California" which were 0.003 in thick. These four bags were felt to cover the extreme cases of the majority of what would be found on the side of the road, so that if these could be successfully picked up, any cases in between would also be successfully picked up.

6.7.2 Stationary Testing Procedures

To set up the stationary testing, the test bed from section 5.2 was used as a mount and the DRA was attached to it so that the height to the ground would be similar to when it was attached to a vehicle. To actually commence the testing, the shuttle was lowered to 2-4 inches above the ground with the jaws completely open. From there, a bag of debris was placed with the top of the bag vertical and the side of the bag fairly centered to and anywhere from touching to 4 inches away from the back plate of the shuttle. The tested height of the shuttle and placement of a bag were felt to mimic that of what was an operator/driver of the vehicle would be able to achieve when the DRA would be attached to a vehicle and used. Once the bag was placed, the jaws would be closed on the bag with the operator attempting to adjust the jaws shutting as he saw fit. This would replicate an operator having to adjust the rate at which each jaw closed to optimally grab a bag of debris successfully, as each bag has different contents and shape. After the jaws gripped the bag, the shuttle was raised to ~6 feet above the ground and weight was applied to the bottom of the bag to see how much force it would take to disengage the bag from the DRA. The method to determine the weight applied before disengagement was having a human stand on a digital scale, tare the scale, then grip the bottom of the bag with a pair of rubber-coated vice grips, and pull down. During the entire process, the human can read the scale readout and see the largest weight change according to the scale, which is equal to the force applied downward by pulling on the bag. Ideally it was desired for the bag to not come loose at all from a human pulling on it (as this would simulate an absolute worst case scenario out in the field), but a more realistic goal was to have the bag not come loose with 25 pounds of force applied. The 25 pound standard would take into account for any kind of large bumps the truck would drive over while the bag was still being lifted up into the truck bed. Lastly, if the bag did not come loose, the shuttle would finish its motion and attempt to deposit the bag by opening the jaws. To have a jaw design that was deemed sufficient, the testing plan as outlined above would have to be implemented on each of the 4 bags outlined in 6.7.1 with at least a 97% (39 out of 40 times) success rate of having a bag withstand the 25 pounds of force and then be deposited into the "truck bed" without being stuck hanging on the jaws or incurring structural damage to the bags of debris (holes in the bags).

6.7.3 Stationary Testing Outcomes and Iterative Designs

Using the stationary testing procedures outlined above, and iterative process was undertaken to improve upon the initial design of the DRA. Figure 2.7 shows the initial design iteration, which is almost exactly the same as in Figure 6.9. The only differences are slightly relocated gearboxes and minor refinement to the shuttle and track, leaving the jaws (or clamshells) the same.

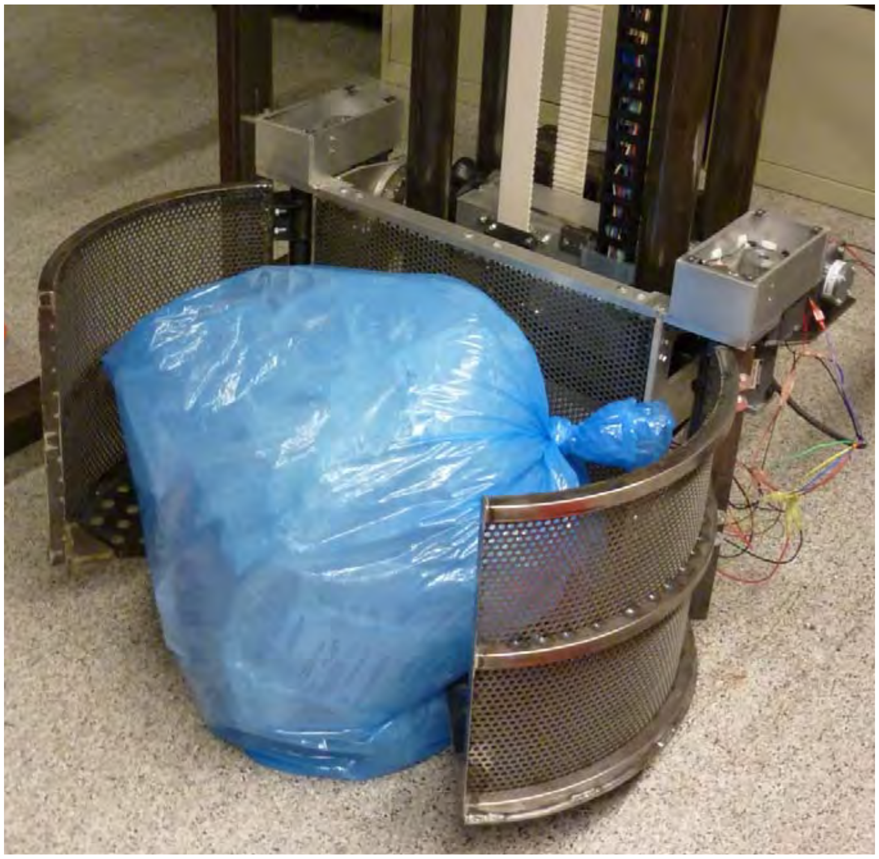


Figure 6.9: Clamshell Design Iteration #1

Design Iteration #1

This first iteration of the clamshells was meant to not only apply a positive pressure onto the bags, but also get underneath the bag and lift it up from the bottom. It was found that getting the clamshells underneath the bag was not a realistic expectation as an operator would not be able to get close enough to the ground and use the DRA without damaging it. This led to the bag being "pinched" by the inner corners of the clamshells and having those be the sole points of contact.

Having two points of contact led to the successful retrieval of bags to vary greatly depending on the individual contents of the bag, and where the two corners grabbed in relation to the contents. If there was something that was not able to move inside the bag as the contents were shifted and the corner was able to get underneath it, the tendency to successfully lift the bag. Many of the bags however incurred structural damage as the two points of contact on the bag would have a large amount of force applied over a small area. Beyond this, the bags would get caught on same corners of the clamshell when trying to be dumped over the top, resulting in the bag not falling as it should. Since the successful outcome of each individual bag could not be predicted and the design didn't meet the criteria listed above, the design was considered unsuccessful and a modified version was created and tested.

Design Iteration #2

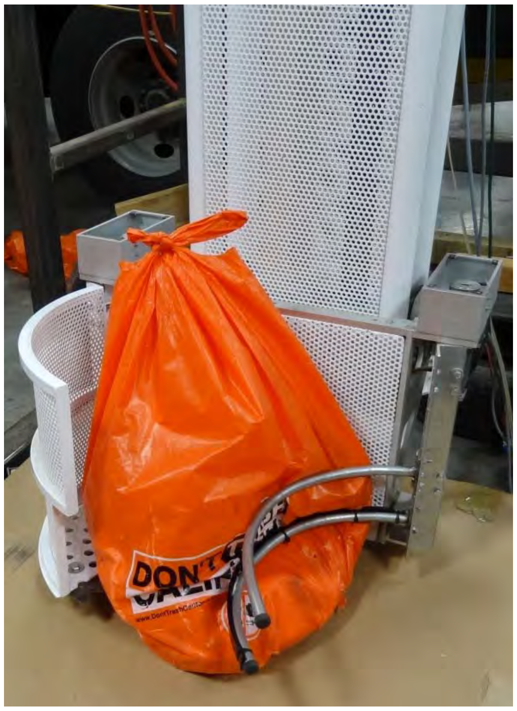


Figure 6.10: Clamshell Design Iteration #2

The iteration shown in Figure 6.10 is the second iteration in the design process of the DRA clamshell. The large design change was that the front clamshell would turn into "pusher arms" and force the bag up onto the flat base of the clamshell. By having bars curved in a concave manner, the point of contact on the bag would always be applying force to the bag toward the back wall or the platform of the other clamshell. This was intended to help aid the successful retrieval of the bag as more of the bag would be lifted from the bottom, reducing the likelihood of shifting contents affecting the ability of the DRA to retrieve a bag of debris successfully.

Having the arms apply the positive pressure (push) to the bag throughout the process and vertical travel was also thought to act as a wall against the bag, further helping slipping from the jaws. During actual use on a vehicle, it was thought that the rear clamshell could stay open partway while the front "pushers" move completely out of the way (Figure 6.11). This would allow the vehicle to use the forward motion having the twofold effect of helping to scoop the bag up onto the back jaw, and keep the vehicle in motion (reducing time needed to finish debris removal).

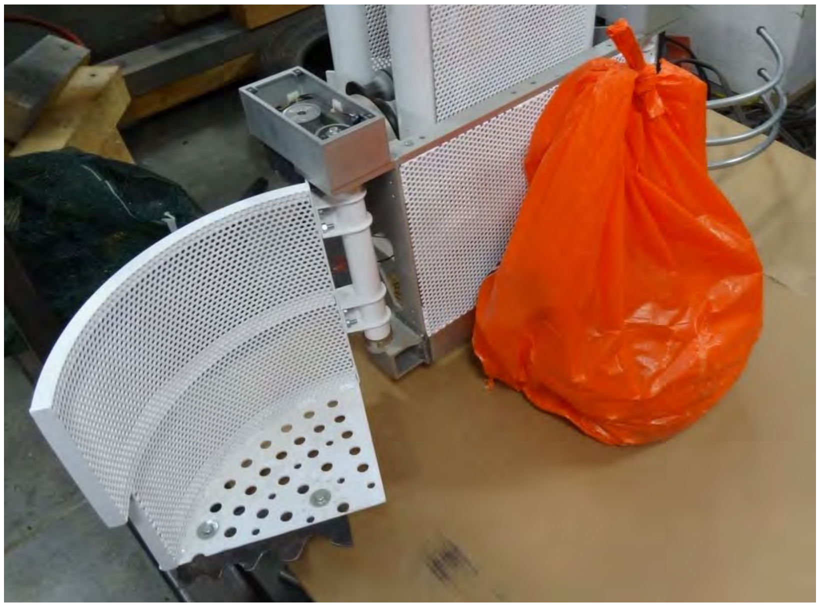


Figure 6.11: Configuration used for scooping bags of debris

This design turned out to have the same issues as design #1 though. The bag of debris tended to get stuck on the same corner of the rear clamshell and not get up onto the flat base of the clamshell. Although the pusher arms helped to add a positive pressure to the bag and hold them tighter, bags still ended up slipping out of the jaws and falling too often. Rubber strips were applied to the pusher arms in hope that it would add more friction keep the bag up during the ascent, but shifting contents made the design too unreliable still.

Design Iteration #3

The final iteration was much like iteration #1, but instead of trying to get under the bag we wanted to grip the bag as evenly as possible in a large number of places. This led to what is shown in Figure 6.13.

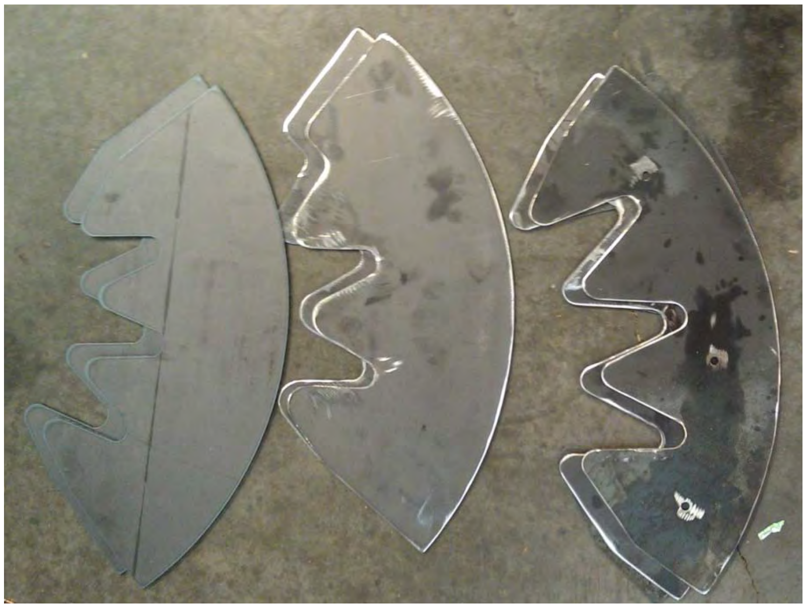


Figure 6.12: Multiple Tooth Plates before installation

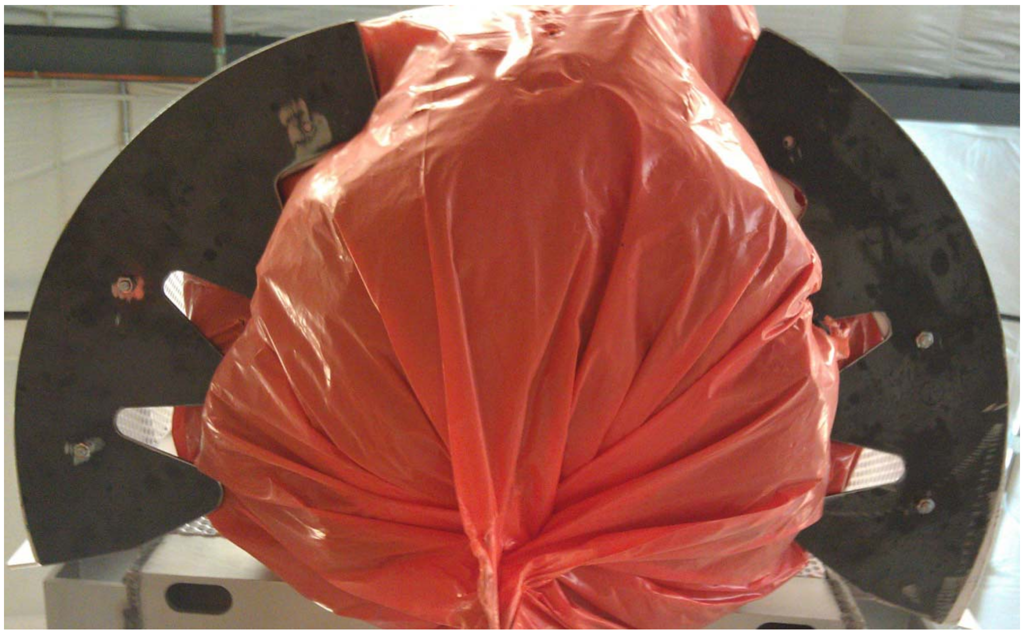


Figure 6.13: Tooth Plate Clamshell Base installed

Multiple tooth plates were manufactured varying the diameter of the location of the teeth points (Figure 6.12) and the number of teeth. This design turned out to be much more successful than previous iterations over all 4 bag types. Only occasionally were bags dropped due to slipping and this was easily rectified by placing another row of teeth along the back wall of the shuttle. The two clamshells closing in on the bag would effectively draw it up against the back teeth while both clamshell teeth dug in. This can be seen in Figure 6.14.

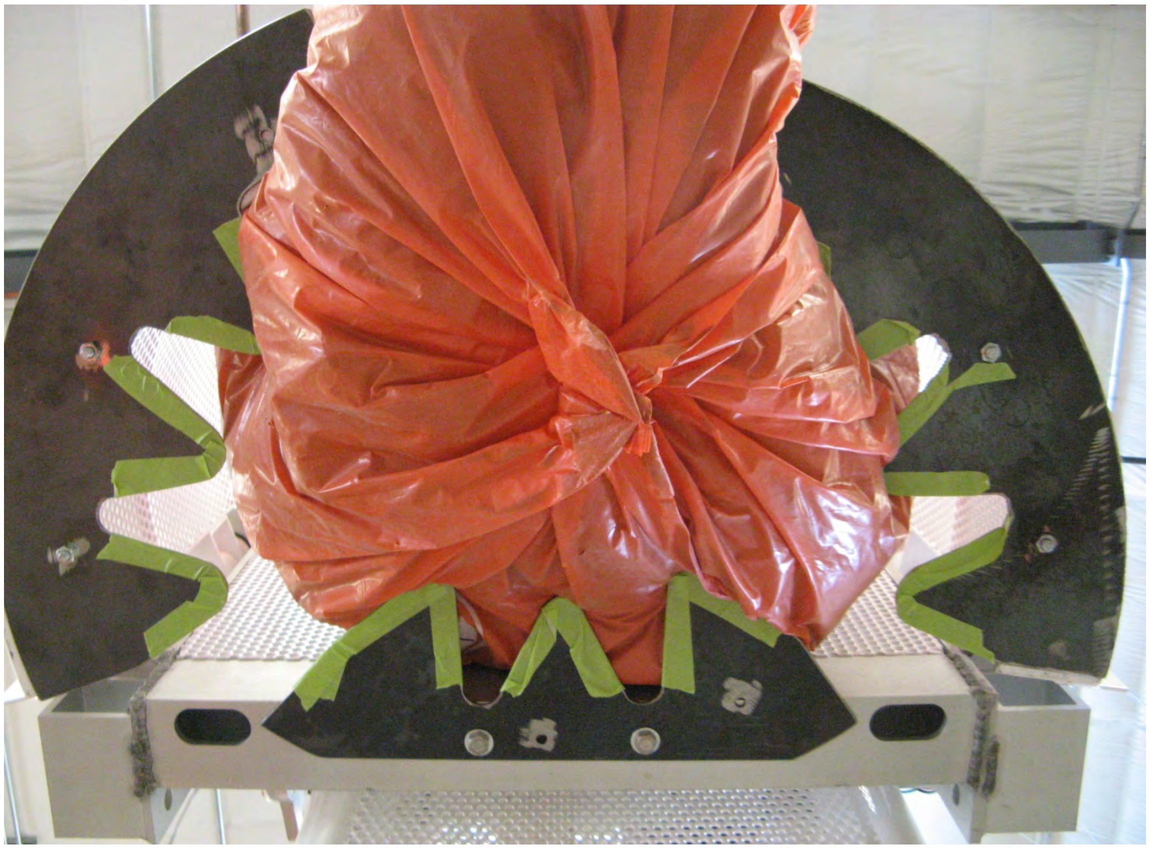


Figure 6.14: Back Tooth Plate installed

Having the multiple points of contact helped to alleviate the problem of shifting contents by supporting enough of the bag to not have shifting affect the outcome of a retrieval operation. Also, the bags very rarely were caught on the clamshells when the dumping action was being taken and when it did get caught, they were shook free from the jaw quickly and easily. It is also of note that the bags did not suffer structural damage from this design iteration.

Preliminary Road Test

The DRA was attached to a 3/4 ton flatbed truck in the standard available configuration and a controlled test was performed with a user operating both the vehicle and the DRA attempting to retrieve the same bags of debris as describe in Section 6.7.1 from the side of a typical road with typical vegetation (Figure 6.15).

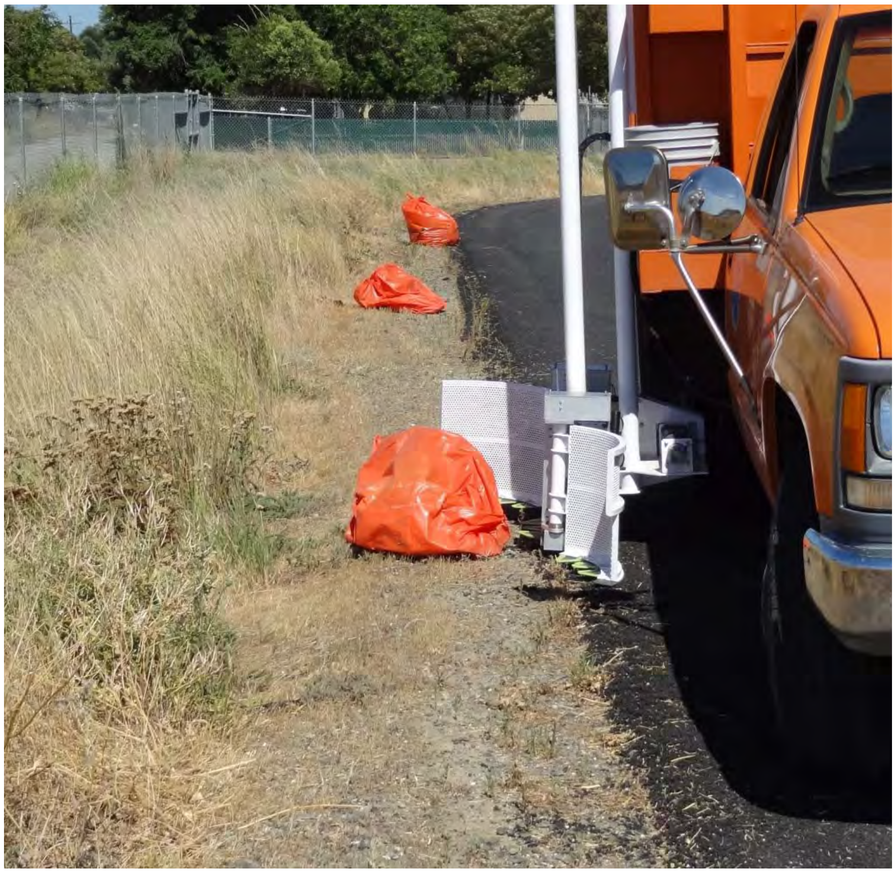


Figure 6.15: Controlled Roadside Testing

The test was done using two people; one placed bags of debris while an operator stayed in the vehicle and concentrated on driving and operating the DRA. The bags were the same kind of bags outlined in 6.7.1 and were attempted to be placed in the same manner as they are found in real world scenarios. This meant that the bags were placed generally upright as they're supposed to be, but not adjusted if they rolled slightly or had their contents shift after being placed. This also entailed the bags being spaced at large enough intervals such that the driver and operator of the DRA could have sufficient distance to align the vehicle to the bag. The operator of the truck and DRA would then drive and attempt to complete a cycle of the DRA, completely picking up and depositing the bag of debris.

The operator initially had a difficult time in gauging where the bags of trash were in relation to the DRA, but after ~1 hour was able to repeatedly successfully pick up bags of debris. Being able to do so was still difficult and took a lot of concentration to ensure that the DRA shuttle did not touch the ground, and was lined up correctly in reference to the bag. To line it up correctly, the operator needed to slow down the truck and barely idle forward while trying to gauge the location of the bag (with reference to the front and back of the truck) using the array of standard mirrors. Adjusting the fisheye mirror enabled the operator to see the location of the bag easier, but correcting for the distortion from the mirror was difficult and getting used to it for long term use seemed like it would also be difficult or potentially not possible. Another point of note is that there was no way to see when the dump cycle had been completed or if the bag was stuck. The operator had to listen for the noise that would occur when a bag was dropped onto the truck bed. This was not a large issue with a mostly empty bed, but could become one when bags of debris are piled up.

Testing the DRA showed that bags were mostly successfully being picked up from the ground, but were getting caught on the clams and not falling into the truck during the dump cycle. The hanging up of the bags was seen to be more than in the stationary lab testing, and the percentage of bags unable to be successfully picked up off the ground was seen to be less than in stationary lab testing. The higher rate of being caught on the clams during the dump cycle was determined to be caused by wind factors that were not seen in a closed building, and that the road/shoulder/median is not level. The lower successful pick up percentage was determined to be mostly caused by the ground being uneven under the bag's location. With the ground's unevenness, it was seen to be difficult to adjust the clam so it could wrap around the bottom of the bag without hitting the ground sometimes. Adding in the truck not being level, and in unique locations such as small ditches, the clamshells could not grip enough of the bag without potentially dragging the DRA on the ground.

As additional testing of the system was performed over additional days, the operator was able to more readily maneuver the truck as needed to capture the bags. Operating the system on the actual roadways with Caltrans crews will be required to properly evaluate the system and determine the ideal modifications necessary. A series of recommendation is discussed in the following chapters.

CHAPTER 7: CONCLUSIONS AND RECOMMENDATIONS

7.1 Introduction

This chapter summarizes the features of the control system and provides recommendations for further development of the complete system. The first section provides an overview of the DRA's electrical design performance, and a discussion of how the prototype compares to the system goals and design parameters. This aspect of the design is critical to the goals of lowering cost and using 12V DC power from the vehicle. Following sections present recommendations for improvements to mechanical components and the complete system. The final section compares the DRA to the original design goals and lists the necessary steps for future design development.

7.2 Control System

A 12V DC electrical system was developed that successfully powers and controls the mechanisms of the prototype Debris Removal Attachment. The basic sensor array gives the minimal configuration to allow the DRA to function without interference. The logic control was provided by an OX2 board, which also has a low power dual motor controller board for the clam motors. An AX3500 high powered motor controller, which receives commands from the OX2, was selected to power the vertical and rotation motors. For the user interface, a PlayStation 2 controller was used with the intention of eventually going wireless. The electronics are all housed in a NEMA 4X enclosure and connects to the DRA with Amp connectors.

The control software was written in C and monitors the configuration of the limit sensors, reads the PS2 controller, and issues motor commands appropriately. The control software will recover from a disconnected or non-functional PS2 controller and can also detect an invalid limit sensor configuration. The operator can be alerted through the use of the PS2 controller's vibration function when an error has been encountered. The software is designed to incorporate additional limit sensors as they are installed in further visions of the DRA.

The clam motor performance was considerably close to the calculated performance, with closing forces of 102 N (23 lbs) and 93 N (21 lbs) for the left and right clams respectively. The initial performance of the DRA shuttle is adequate, with lift times of 9.48, 9.94, 11.36 seconds for unloaded, 4.5 kg (10 lbs), and 13.6 kg (30 lbs) trials respectively. The lift times were considerably higher than the calculated times, which ignored friction. The rotation times for the same trials were 2.13, 2.28, and 2.59 seconds respectively. These times very closely represented the calculations for the system, as friction seems to play a much smaller part in the rotation system. The DRA performance gives dump cycle times between about 11.5 and 14.0 seconds for the unloaded through 13.6 kg (30 lbs) cases. These lift times should improve as work is done to reduce friction and binding issues in the shuttle path.

7.3 Recommendations for Controls System Improvements

There are several recommendations that will help make the system more robust and effective. These recommendations include improvements and add-ons such as a dedicated custom circuit board, high flex multi-conductor cable, and additional sensors. Also included in this section are features that were originally intended to be implemented in the DRA, but because of complications and in the interest of time were put aside. Their description, benefit, and suggestions on how to implement them are described below, and include wireless PS2 communication and RS232 serial communication.

7.3.1 Dedicated Custom Circuit Board

A custom circuit board could be constructed to provide the function of the 5V regulator and the dual channel brake relay. The board could also add some new features, such as opto-isolators for each of the IO pins of the OX2. Opto-isolators provide electrical isolation for sensitive electronics, such as the OX2, by using an internal LED to switch a phototransistor [41]. No direct electrical connection exists between the LED and phototransistor, so a power spike on one side cannot travel to the other. A separate relay could also be installed, which could be actuated by the OX2 to switch power to the AX3500's logic control, which is separate from the AX3500's motor power (see Section 5.6 Error Protocols). Other features could be integrated into the circuit board.

The custom circuit board would therefore add functionality and increase the robustness of the electronics while simultaneously reducing the overall circuit board footprint of the DRA. Reducing the circuit board footprint could help allow the use of a smaller NEMA box, which would be easier to mount on a truck.

7.3.2 Multi-Conductor Flex Cable

It is recommended that the discrete cables running through the cable carrier to the NEMA box be replaced with high flex multi-conductor cable. Multi-conductor cable will clean up the appearance of the wires make it easier to rout and track wires during assembly and maintenance. Wires for specific subsystems or of similar purpose, such as clam motors, clam sensors, etc., can be grouped together, or systems with similar power requirements can be grouped together. High flex cable is intended to be used in systems that involve periodic or frequent bending, such as cable carriers [42]. This will ensure that the fatigue life of the wire shielding will not become an issue. Careful attention should be paid to the current carried on each line when selecting the proper gauge, since the requirements vary greatly for different wires.

7.3.3 Additional Sensors

As the system stands now, there are no sensors to detect the bottom and docked limits of travel. Without the sensors, the unit relies on the operator to be aware of the bottom of travel and to stop the shuttle motion appropriately. If the operator is not aware or not paying enough attention, the vertical motor could become stalled for an unacceptable amount of time and damage to the shuttle could occur. Though the AX3500 will protect itself by automatically limiting the current if the drive electronics become too hot, it is still undesirable for the motors to stall and should be avoided.

Adding these sensors will complete the DRA's limit sensor array. Although the system should be able to function adequately without these sensors, the DRA will be more robust with them. The control software is setup to incorporate these sensors, though currently their use is disabled. Small changes to the control software will allow these sensors to be integrated into the system.

If needed, non-limit sensors could be added to the sensor array, such as temperature sensors for the motors or electronics. With some rearranging of the IO ports of the OX2, there are unused IO lines that can allow for analog sensors as well.

Also, a height sensor of some sort to let the operator know when the shuttle was close to the ground would reduce the likelihood of damaging the DRA. The sensor could either give feedback to the operator through lights or to the controller's vibration capabilities, or initiate an automatic stop when close to the ground. Whether the sensor would be a mechanical switch or a microcontroller operated piece is yet to be determined as more research would need to be done to decide on an appropriate one.

7.3.4 RS232 Communication

The AX3500 is capable of communicating via RS232 serial [13]. This allows for more complete and effective communication between the OX2 and the AX3500. With RS232 serial, precise motor commands to be sent, control parameters can be changed such as current limiting, and data can be read back such as the current levels in the high power motors. With the ability to monitor the current levels in the high power motors, the OX2 could have two additional features. The controller could limit the current in the vertical motor when the shuttle is traveling down, which would prevent the motor from stalling due to an obstacle such as a curb. A second feature that could be added is the ability to detect and prevent overloading the shuttle. Since the current in the vertical motor is proportional to the load on the shuttle, the OX2 could detect if the shuttle was overloaded and stop motor commands and alert the operator.

7.3.5 Replace NEMA Box Connectors

The connectors used to connect to the NEMA box should be replaced. There are two shortcomings of the present connectors: (1) they are not water tight, and (2) they are incorrect orientation such that the male pins are the energized pins. Male pins that are energized risk being shorted unintentionally if the mating side is not connected. The connectors could not simply be reversed since the mating side that disconnects from the connector must be on the outside of the NEMA box. New connectors should be used to provide a sufficient level of protection from water, and with reversed orientation such that the disconnect side contains the male pins. The connectors should also be rated at adequate current levels.

7.4 Recommendations for Improved DRA Operation

7.4.1 Vision System

Employing a closed circuit camera system of some sort near the bottom of the shuttle guide rail facing towards the front end of the vehicle would enable an operator to see a bag of debris approaching the DRA giving a better spatial awareness and most likely improving the ease of use and success rate. These systems are very inexpensive to implement and for the low cost a significant change in ease of use would occur.

7.4.2 Curved Teeth on the Shuttle Wall and Clamshells

Having teeth that curve slightly upwards towards the top of the shuttle will help prevent hang-ups of the bags during the dumping operation. The teeth would point upwards during the bag's ascent helping the teeth grip even more. At the dumping orientation the teeth are pointed down and will release the bags more readily.

7.4.3 Enhanced Truck Mirror Array

In addition to the vision system, a set of mirrors would help visibility during all sections of the DRA's operation and the vehicle's normal use. Using a normal set of side mirrors on a 3/4 ton flatbed truck in the standard available configuration, one can only see the retrieval operation at the clamshells. One the ascension has begun you cannot view the shuttle or the bag, forcing the user to rely on hearing the noise of a bag falling into the truck bed, which during highway operation could be difficult and unreliable. Since the DRA also extends from the side of the vehicle, vision to the rear of the right side of the truck is impaired significantly. An appropriately modified mirror array would help to alleviate this also.

7.4.4 Bag Retrieval Success Rate

The DRA worked well, completing its goal of successful and repeatable debris removal. The bag retrieval success rate was above 90% but a rate of over 97% is recommended for the final product. All refinements that improve the collection success rate should be implemented before extensive field testing.

7.4.5 Mechanical System

The mechanical components of the system proved to be reliable and robust in testing to date. Several design changes will be required before full implementation. The method of attachment to the vehicle must be revised to make it simple and quick to install and remove. This will require a permanent installation of brackets to the Caltrans vehicles. Additionally, a cart will have to be designed to roll the system to and from the truck so that it can be installed without the use of a fork lift as is presently done.

The fact that the system extends beyond the width of the bed may be problematic. To retract the system within the 8ft width of the truck bed would require changes to the truck bed and/or a complete redesign of the DRA. Closely related to this is the need to determine a range of DRA heights that would be required to support the typical fleet. These changes will incur various costs and must be considered as part of the final assessment.

7.5 Meeting Design Goals

The DRA design meets most of the design goals listed in Chapter 2 as applied to the litter bag collection operation. The concept will increase worker productivity and safety.

As demonstrated it will keep workers in the vehicle most of the time. Periodically the operator will have to exit the vehicle to load some of the items found next to bags that cannot be grasped by the clam shell. Typically the operator would manually place the odd items into the clam shell and then operate the DRA as usual.

The DRA is simple to operate and the control system can readily be changed to incorporate various automated functions that can make it even easier to operate.

The goal to develop a low cost system has been mostly achieved. The control system design makes use of consumer market components, which can greatly reduce costs. The estimated cost for a DRA system in a final configuration is expected to be less than $15000.

The goal of a simple and minimal machine design that is easy and inexpensive to purchase and maintain has been potentially met by this concept. It demonstrates the simplest configuration of a mechanism to successfully grasp and lift a bag into the truck bed.

With minor modifications to typical trucks found in the Caltrans fleet, the DRA could be integrated into the current fleet as an attachment not requiring dedicated vehicles. These modifications would have minimal impact on other vehicle functions.

By powering off the 12V DC system of the truck, the DRA would not require the implementation of a hydraulic power take off or the use of a separate small engine. This will significantly reduce costs and maintenance requirements. Additionally, avoiding the use of a separate engine will reduce pollution.

Prior to further development, information from Caltrans will be required for a complete assessment of the concept's viability. Crews must operate the system in typical scenarios and then provide the user feedback necessary to identify design limitations. Impacts on the litter collection process need to be identified. At a minimum, bags would have to be placed such that the DRA can easily access them. A determination of need must be made to establish the potential number of units that would be required. With this information a cost benefit analysis can be performed to decide what the final DRA configuration would be. Given the Caltrans experience with the DRV and the development of the DRA, the widest range of automation solutions to the litter bag retrieval challenge has been demonstrated.

REFERENCES

[1] Andres, D. L.. “Disposal of Roadside Litter Mixtures.” National Cooperative HighwayResearch Program: Synthesis of Highway Practice. Washington, D.C.: National Academy Press, 1993.

[2] Caltrans. *Caltrans Litter Abatement Plan*. N.p.: Caltrans, 2007. Web. 22 Jun. 2009 <www.dot.ca.gov/docs/LitterAbatementPlan.pdf>.

[3] "Maintenance Department- Litter." *California Department of Transportation*. Web. 25 July 2009. <http://www.dot.ca.gov/dist11/departments/ mainten/litter.htm>.

[4] "DebrisRemoval - AHMCT." *Main Page - AHMCT*. Web. <http://ahmct.ucdavis.edu/index.php?title=DebrisRemoval>.

[5] "Caltrans Observes Highway Worker Memorial Day." *California Department of Transportation.*, 25 Oct. 2009. Web. 3 Nov. 2009 <http://www.dot.ca.gov/dist07/Publications/Inside7/story.php?id=107>.

[6] "Caltrans Worker Hit, Killed On Highway 99 In Lodi." *CBS*. 23 Jul 2009. Web. 25 Oct 2009. <http://cbs13.com/local/Caltrans.Worker. Killed.2.1098766.html>.

[7] Burkett, George. *Debris Removal Vehicle*. Tech. Davis: AHMCT, 2009. Print.

[8] "Cable Carrier Kits." *Gleason Reel Corporation*. Web. <http://www.hubbell-gleason.com/preeng/trakkits.html>.

[9] *Sliprings*. Orbex Group. Web. <http://www.slipring.com/miniaturecapsule.html>.

[10] *Digital Vane Switch*. Cherry Corporation. Web. <http://www.cherrycorp.com/ english/sensors/Magnetic_Proximity/vn1015.htm>.

[11] *Plastic Threaded Barrel*. Hamlin. Web. <http://www.hamlin.com/product-detail.cfm?productid=52>.

[12] *CPI*. Control Products Inc. Web. <http://cpinj.thomasnet.com/item/waterproof-switches-limit-switches/e-series-limit-brackets/e1101?&seo=110&plpver=10>.

[13] "AX3500 Product Folder." Roboteq Inc. Web. <http://www.roboteq.com/ax3500-folder.html>.

[14] RoboteQ. *RoboRun*. Computer software. *RoboteQ*. Vers. 1.9d. 15 Feb. 2008. Web. <www.roboteq.com>.

[15] "Orangutan X2 Robot Controller Quick-start Guide." *Pololu* Pololu, n.d. Web. <http://www.pololu.com/file/0J43/ox2_quickstart.pdf>.

[16] "Katana- PlayStation 2 Wireless Controller." *Amazon*. Web. <http://www.amazon.com/Playstation-2-Wireless-Controller-Silver/dp/B000WE8JCA/ref=sr_1_2?ie=UTF8&s=videogames&qid=1257293138&sr=1-2>.

[17] "Logitech- PlayStation 2 Wireless Controller." *Amazon*. Web. <http://www.amazon.com/gp/product/B0001VNNE8/ref=s9_simz_gw_s0_p63_i2?pf_rd_m=ATVPDKIKX0DER&pf_rd_s=center-2&pf_rd_r=09ASEP6DFQ23P8E321WG&pf_rd_t=101&pf_rd_p=470938631&pf_rd_i=507846>.

[18] "Dual Hi/Lo Switched Relay." *Frys Electronics*. Web. <http://www.frys.com/product/4986031?site=sr:SEARCH:MAIN_RSLT_PG>

[19] *Power-One.com*. Web. <http://www.power-one.com/>.

[20] "ATmega644- Product Card." *Atmel Corporation*. Web. 03 Nov. 2009. <http://www.atmel.com/dyn/products/product_card.asp?part_id=3694>.

[21] Atmel. *AVR Studio*. Computer software. Vers. 4.16. Atmel Corporation, 2009. Web. <www.atmel.com>.

[22] *WinAVR*. Computer software. Vers. 20090313. Open Source, 2009. Web. <http://winavr.sourceforge.net/>.

[23] *Pololu Robotics Forum*. Pololu. Web. <http://forum.pololu.com/>.

[24] *AVR Freaks*. Web. <http://avrfreaks.com>.

[25] *Groschopp Integrated Motion Solutions*. Web. <http://groschopp.com/>.

[26] Karnopp, Dean C., and Donald L. Margolis. *Engineering Applications of Dynamics*. New York: Wiley, 2007. Print.

[27] Hollis, Selwyn. *Differential Equations with Boundary Value Problems*. Upper Saddle River: Prentice Hall, 2002. Print.

[28] Das, Shuvra. *Mechatronic Modeling and Simulation Using Bond Graphs*. Boca Raton: CRC, 2009. Print.

[29] *Harmonic Drive Gearing*. Harmonic Drive LLC. Web. <http://www.harmonicdrive.net/>.

[30] *Matlab*. Computer software. Vers. 7.0.4. The MathWorks. Web. <www.mathworks.com/>.

[31] *Pro/Engineer*. Computer software. Vers. 4.0 Wildfire. PTC. Web. <http://www.ptc.com/products/proengineer/>.

[32] "Interfacing a PS2 Playstation Controller." *Curious Inventor*. Curious Inventor LLC. Web. <http://www.curiousinventor.com/guides/ps2>.

[33] "Orangutan X2 Resources." *Pololu Robotics and Electronics*. Pololu Electronics. Web. <http://www.pololu.com/catalog/product/738/resources>.

[34] "RS-232." *Wikipedia, the free encyclopedia*. Web. <http://en.wikipedia.org/wiki/RS-232>.

[35] "Terminals - Quick Connects, Quick Disconnects." *Digi-Key Corporation*. Web. <http://parts.digikey.com/1/parts-cats/terminals-quick-connects-quick-disconnects-connectors>.

[36] "NEMA Enclosure Types." *NEMA - National Electrical Manufacturers Association*. Web. <http://www.nema.org/prod/be/enclosures/upload/ NEMA_Enclosure_Types.pdf>.

[37] "NEMA Enclosures." *McMasterr - Carr*. McMaster - Carr Supply Company. Web. <http://www.mcmaster.com/#electrical-enclosures/=4czl5n>.

[38] *Cooper Bussmann*. Web. <http://www.cooperbussmann.com/>.

[39] "AMP Circular Plastic Connectors." *Jameco Electronics*. Web. <http://www.jameco.com/webapp/wcs/stores/servlet/GroupDisplay?langId=-1&storeId=10001&catalogId=10001&groupName=AMPCircular>.

[40] *Pelouze*. Web. <http://www.pelouze.com/>.

[41] "Opto-isolator." *Wikipedia, the free encyclopedia*. Web. <http://en.wikipedia.org/wiki/Opto-isolator>.

[42] "Multiconductor Cable." *McMaster - Carr*. McMaster - Carr Supply Company. Web. <http://www.mcmaster.com/#catalog/115/785/=4czt4w>.

APPENDIX A: MATLAB AND SIMULINK CODE AND RESULTS

```
%DRA_Rotation_Time_run.m
%Purpose: To calculate the rotation time of the shuttle's dump assembly.
%Also calculates the current draw and the energy used.
%This code automatically initializes and calls the Simulink file
%DRA_Rotation_Time.sim and produces plots.

%BEGIN: User Inputs
motor_select = 8;                          %select motor 1 through 10
gear_ratio_array = [50 80 100 120 160]; %50, 80, 100, 120, 160 available
%END: User Inputs

close all;
%BEGIN: Geometric Input Parameters
typical_load = 10;          %load in lbs
max_load = 150;             %load in lbs
wcg_en = 87;                %weight of shuttle in lbs
xp_en = 14.0;               %width to payload in inches 14
yp_en = 11.5;               %height to payload in inches 11.5
xcg_en = 4.66;              %width to center gravity in inches 4.66
ycg_en = 7.19;              %height to center gravity in inches 7.19
shuttle_inertia = 4.6056; %in kg-m^2
%END: Input Parameters

%BEGIN: Motor stats
%           no-load speed  stall torque  torque constant  stall current    catalog
%              (RPM)         (lbs-in)      (lbs-in/amp)       (amps)        number
motor_stats = [   3000          6.60           0.33            22.45;...   %1000(1)
                  1350          5.30           0.74            8.150;...   %1006(2)
                  1475          4.10           0.66            7.310;...   %1010(3)
                  3200          5.80           0.31            21.88;...   %1011(4)
                  2700          20.0           0.37            61.82;...   %2000(5)
                  1300          17.5           0.75            26.19;...   %2006(6)
                  2650          31.0           0.38            90.31;...   %4100(7)
                  1325          29.0           0.76            41.79;...   %4106(8)
```

```
                1200        40.0          0.83          54.68;...   %4300(9)
                1350        54.0          0.76          79.24; ];    %4400(10)

%BEGIN: Calculations
no_load_speed = motor_stats(motor_select,1);                    %in RPM
stall_torque_en = motor_stats(motor_select,2);                  %in lbs-inches
motor_torque_constant_en = motor_stats(motor_select,3);         %in lbs-in/amp
stall_current = motor_stats(motor_select,4);                    %in amps

%change to SI units
stall_torque = 0.112984829*stall_torque_en;
motor_torque_constant = 0.112984829*motor_torque_constant_en;
wcg = 4.4481*wcg_en;
xp = 0.0254*xp_en;                  %Payload (subscript p)
yp = 0.0254*yp_en;
xcg = 0.0254*xcg_en;                %Center gravity (subscript cg)
ycg = 0.0254*ycg_en;
phi1 = atan(yp/xp);
phi2 = atan(ycg/xcg);
r1 = (yp^2+xp^2)^(1/2);
r2 = (ycg^2+xcg^2)^(1/2);
rw = 12/stall_current;                                  %Winding Resistance
energy_used = zeros(length(gear_ratio_array),2); %row: weight - column: gear ratio

%Loop for different gear ratios
for i = 1:length(gear_ratio_array)

gear_ratio = gear_ratio_array(1,i);
no_load_speed_gear = no_load_speed/gear_ratio;
stall_torque_gear = stall_torque*gear_ratio;
speed_rad_sec = no_load_speed_gear*pi/30;
friction_torque = motor_torque_constant*stall_current - stall_torque;
slope = -stall_torque_gear/speed_rad_sec;
offset = stall_torque_gear;

wp_en = typical_load;                %weight of payload in pounds
```

```
wp = 4.4481*wp_en;
mp = wp/9.81;
payload_inertia = mp*r1^2;
inertia = payload_inertia + shuttle_inertia;
%END: Calculations


%Begin 1st simulation and plotting
sim('DRA_Rotation_Time_ver2');


energy_used(i,1) = max(energy);


figure(1);
set(gcf,'Position',[464 663 560 420],'NumberTitle','Off','Name','Angle vs. Time')
plot(tout,angle,'-r');
title 'Angle vs. Time';
xlabel 'Time (s)';
ylabel 'Angle (degrees)';
hold on;


figure(2);
set(gcf,'Position',[464 161 560 420],'NumberTitle','Off','Name','Gearbox Torque vs. Time')
plot(tout,gearbox_torque/0.112984829,'r');
title 'Gearbox Torque vs. Time';
xlabel 'Time (s)';
ylabel 'Gearbox Torque (in-lbs)';
hold on;


figure(3);
set(gcf,'Position',[1034 663 560 420],'NumberTitle','Off','Name','Gearbox Torque vs. Angle')
plot(angle,gearbox_torque/0.112984829,'r');
title 'Gearbox Torque vs. Angle';
xlabel 'Angle (Degrees)';
ylabel 'Gearbox Torque (in-lbs)';
hold on;


figure(4);
```

```
set(gcf,'Position',[1034 161 560 420],'NumberTitle','Off','Name','Motor Current vs. Time')
plot(tout,current,'-r');
title 'Motor Current vs. Time';
xlabel 'Time (s)';
ylabel 'Motor Current (amps)';
hold on;
%END: 1st Simulation and Plotting

%BEGIN: 2nd Simulation and Plotting
wp_en = max_load;                    %weight of maximum payload in lbs
wp = 4.4481*wp_en;
mp = wp/9.81;
payload_inertia = mp*r1^2;
inertia = payload_inertia + shuttle_inertia;

sim('DRA_Rotation_Time_ver2');

energy_used(i,2) = max(energy);

figure(1);
plot(tout,angle,':b');
legend('10 lbs','150 lbs',4);

figure(2);
plot(tout,gearbox_torque/0.112984829,'b');
legend('10 lbs','150 lbs',1);

figure(3);
plot(angle,gearbox_torque/0.112984829,'b');
legend('10 lbs','150 lbs',1);

figure(4);
plot(tout,current,':b');
legend('10 lbs','150 lbs',1);
%END: 2nd Simulation and Plotting
end;
```

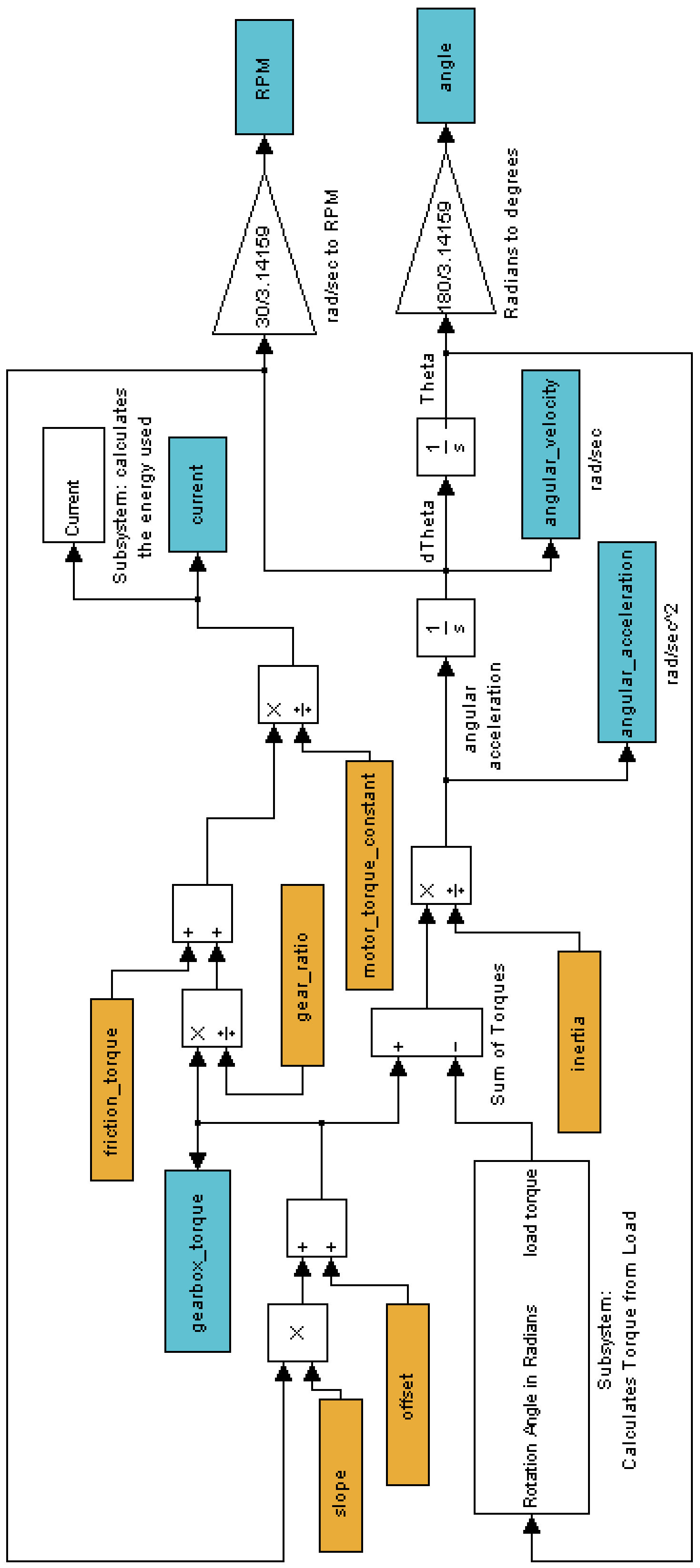


Figure A. 1: Top Level Simulink Code

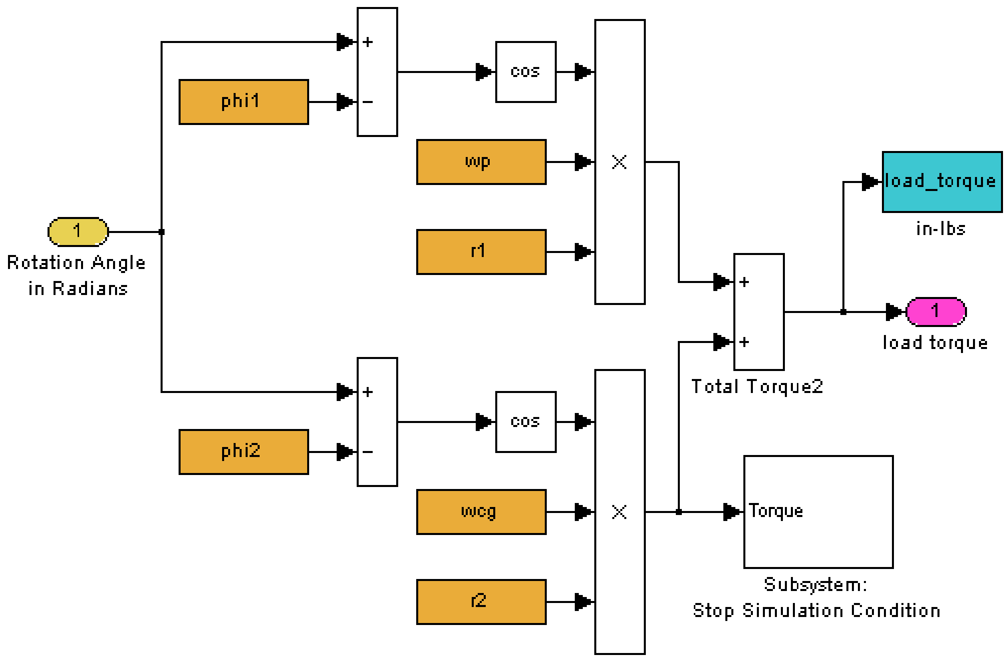


Figure A. 2: Load Torque Subsystem

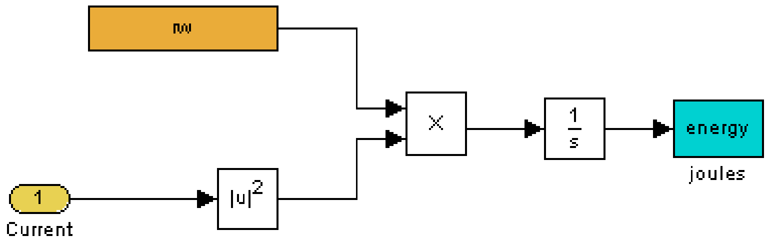


Figure A. 3: Energy Calculation Subsystem

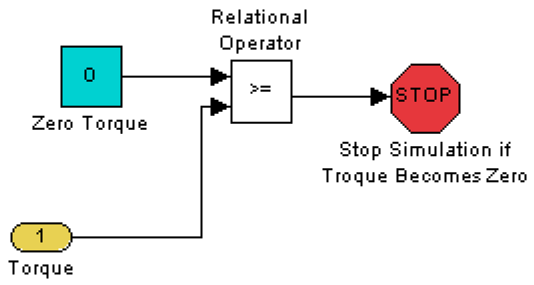


Figure A. 4: Stop Simulation Condition

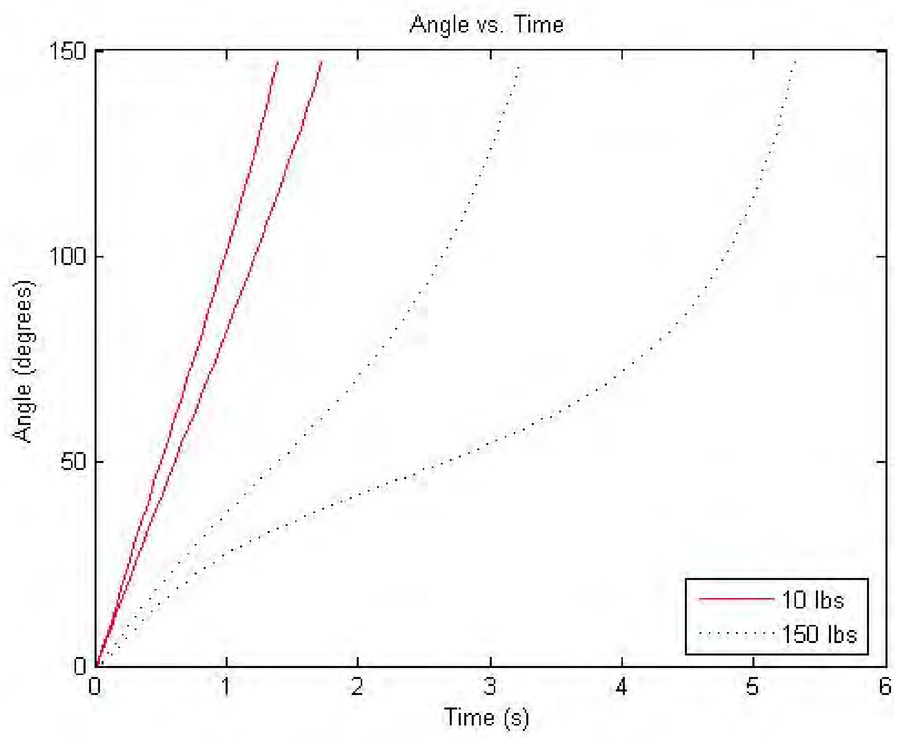


Figure A. 5: Angle vs. Time for Motor 4100 with 120 and 160:1 Gear Ratio

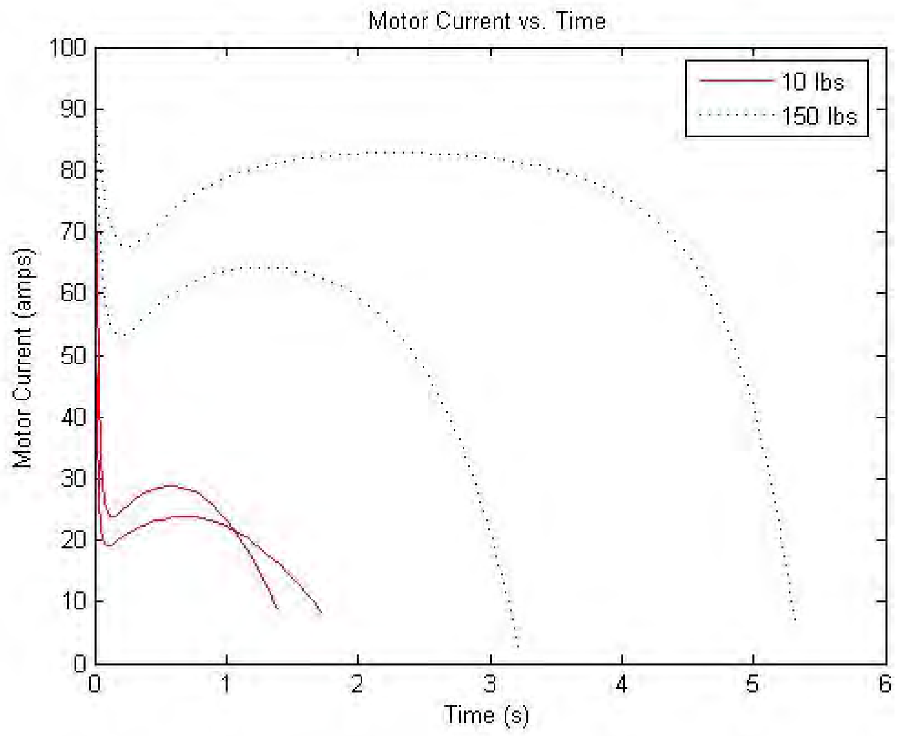


Figure A. 6: Current vs. Time for Motor 4100 with 120 and 160:1 Gear Ratio

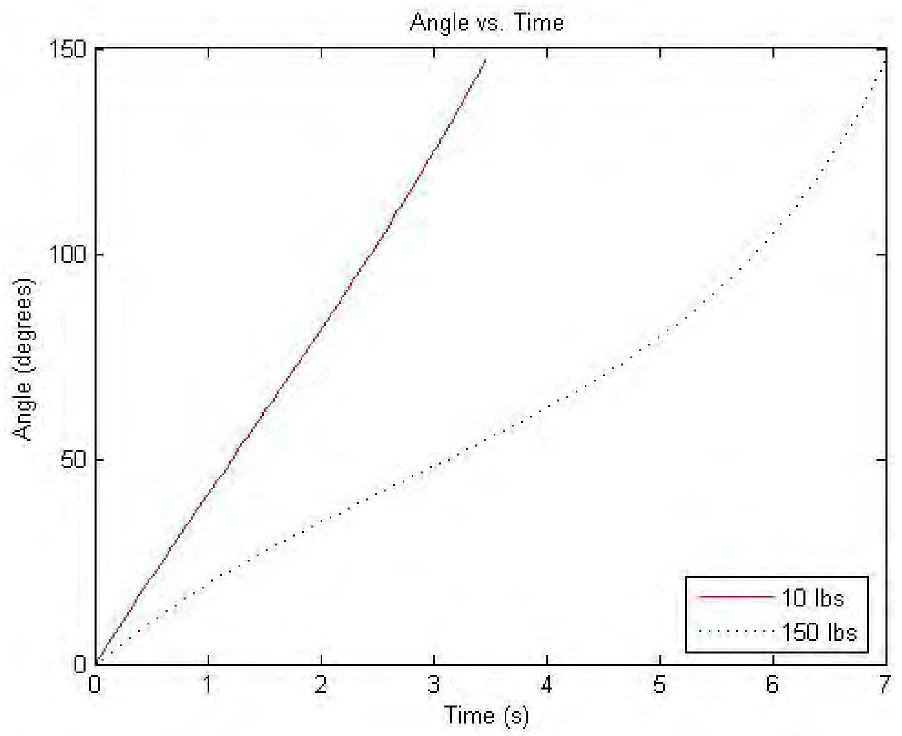


Figure A. 7: Angle vs. Time for Motor 4106 with 160:1 Gear Ratio

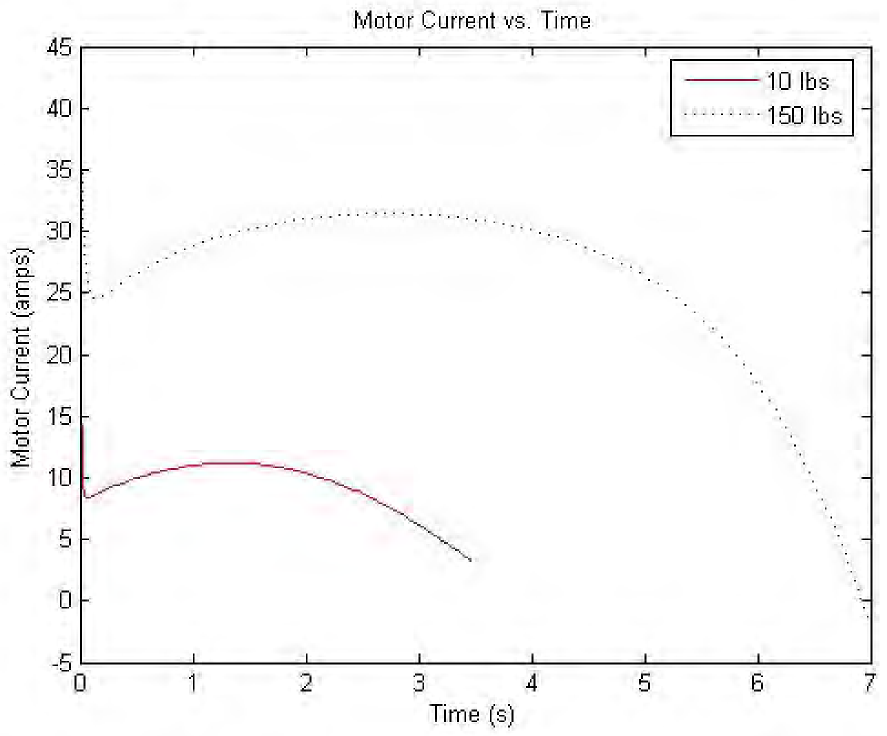


Figure A. 8: Current vs. Time for Motor 4106 with 160:1 Gear Ratio

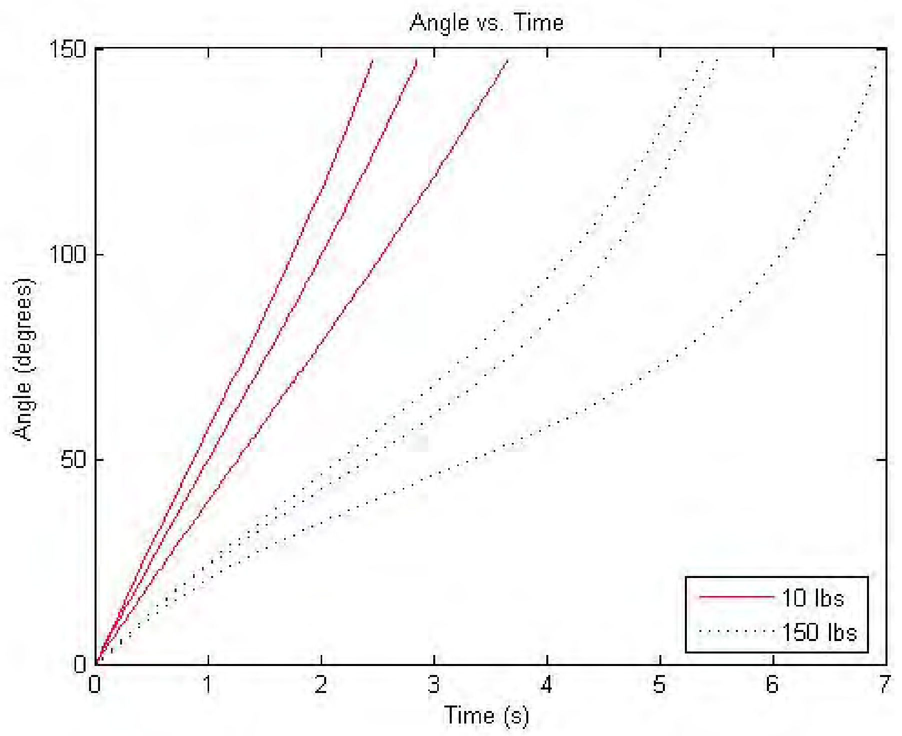


Figure A. 9: Angle vs. Time for Motor 4300 with 100, 120, and 160:1 Gear Ratio

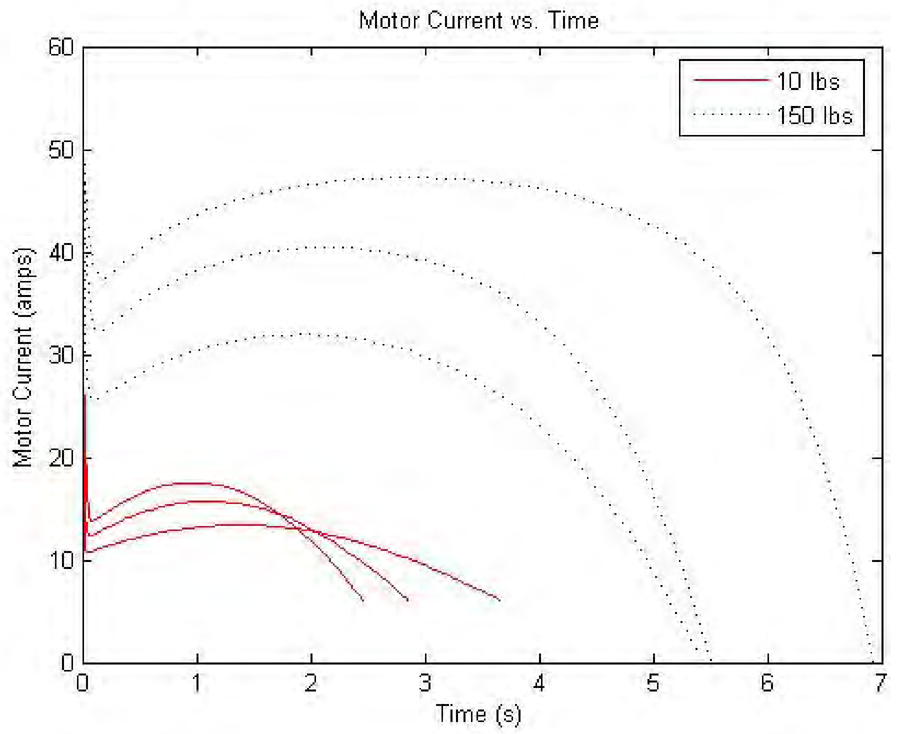


Figure A. 10: Current vs. Time for Motor 4300 with 100, 120, and 160:1 Gear Ratio

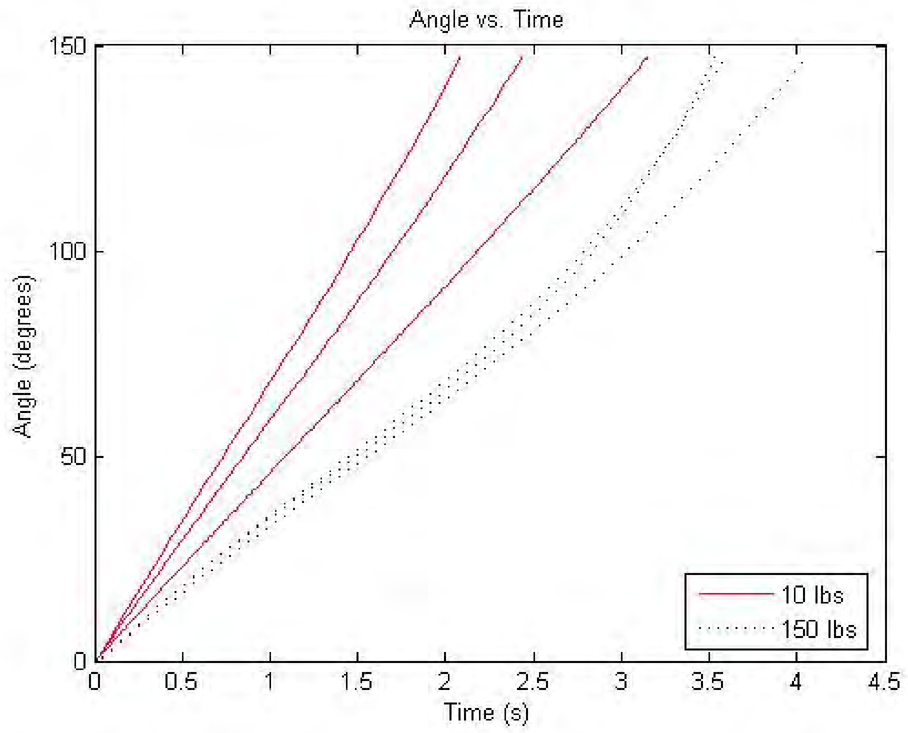


Figure A. 11: Angle vs. Time for Motor 4400 with 100, 120, and 160:1 Gear Ratio

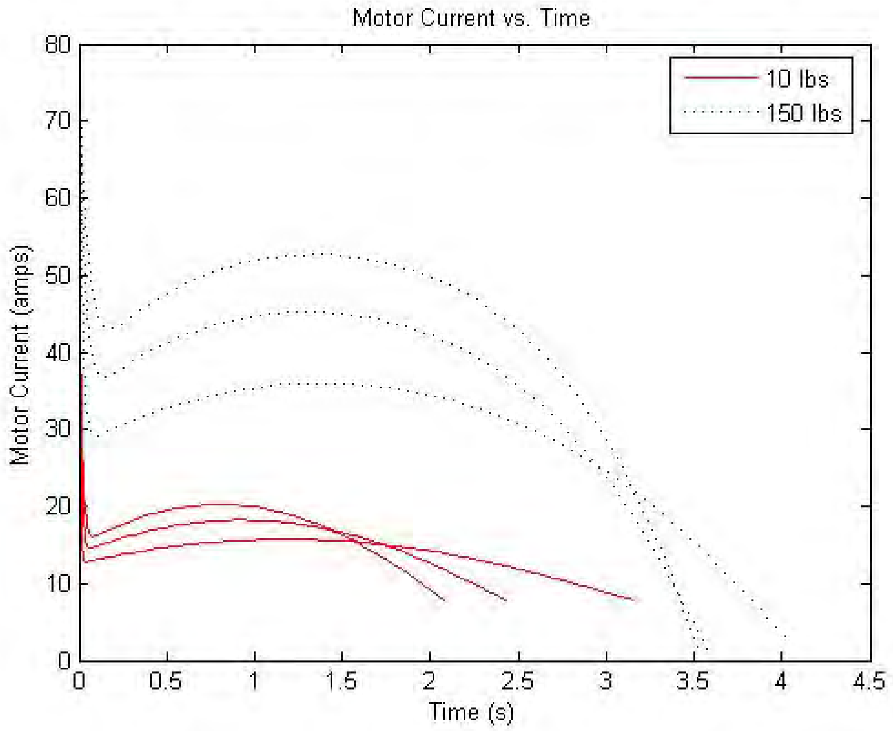


Figure A. 12: Current vs. Time for Motor 4400 with 100, 120, and 160:1 Gear Ratio

APPENDIX B: DRA CONTROL SOFTWARE

```
/***********************************************************
This is the control software for the DRA (DRA_Main.c)
Version 1.0
Last modified:    October 21th 2009
Created:          August 10th 2009
Code by Matthew H. Jones

Notes and Revisions:

***********************************************************/

// BEGIN Main Code

#include<DRA.h>         //DRA.h contains all other include files for
                        //DRA system

// BEGIN Main Function

int main(void)
{
// BEGIN Initialize Utilities

SPIInit();
DRAInit();
PS2Init();
PULSEInit();
//LCDInit();                  //uncomment when needed for debugging
saveM1CurrentLimit( 0x00, 0x00 );   //disable current limit
saveM2CurrentLimit( 0x00, 0x00 );   //disable current limit
// END Initialize Utilities

// BEGIN Initialize Variables

int DRA_ERROR = 0;            //error flag
int Sensors = 0;              //sensor configuration as a binary value
int SmallMotor = 0;           //small motor command (0xFF on, else off)
int LargeMotor = 0;           //large motor command (0x00 - 0xFF)
int LastVerticalBrake = 1;    //stores last vertical brake value, brake
                              //initializes locked
int LastRotationBrake = 1;    //stores last rotation brake value, brake
                              //initializes locked
int ClamInvert = 0;           //if operator is in the truck, need to
                              //invert clam motor signals by setting to
                              //nonzero value
int ClamInvertFlag = 0;       //stores if last loop the controls were
                              //inverted, if so buttons are still held, //do
                              not invert again. This keeps the OX2 //from
                              inverting the controls repeatedly //while the
                              user holds invert buttons
VerticalSpeed  = 0;
RotationSpeed  = 0;
LeftClamSpeed  = 0;
RightClamSpeed = 0;
// END   Initialize Variables

// BEGIN Main Loop

while(~DRA_ERROR) //Loop while there is no DRA_ERROR
{
PS2Get(SmallMotor, LargeMotor); //Read PS2 controller and issue
                                //vibration motor commands
Sensors = ( ~(PIND) & (SENSOR_SHUTTLE_SENSORS) );//Read and store
                                                 //shuttle sensor values

//*****************************************************************
// BEGIN Set the vertical and rotation motor speeds based on shuttle
// configuration

//BEGIN main if statement
if(PS2ID == PS2_ANALOG_MODE)  //only execute if PS2 controller is
                              //present and in analog mode
{
switch(Sensors)   //Input "Sensors", branch based on configuration
{                        //Output is vertical and rotation motor
```

```
                        //velocities
case DRA_TOP_0    :     Top0();
                        break;

case DRA_TOP_180  :     Top180();
                        break;

case DRA_DOCKED   :     Docked();
                        break;

case DRA_BOTTOM   :     Bottom();
                        break;

case DRA_MID_0    :     Mid0();
                        break;

case DRA_MID_180  :     Mid180();
                        break;

case DRA_TOP_MID  :     TopMid();
                        break;

default           :     Error();
                        break;
}
//Need to switch polarity of the vertical and rotation motor commands
VerticalSpeed = -VerticalSpeed;
RotationSpeed = -RotationSpeed;

//R1 is the enable button on the PS2 controller for the vertical and
//rotation drives
if( (PS2CMD2 != (1<<PS2_R1)) )
      {
      //if button is not pressed, zero motor speeds
      VerticalSpeed = 0;
      RotationSpeed = 0;
      }
//*********************************************************************
```

```
// BEGIN temporary section
//This section of code allows the user to override the top sensor value
//and rotate the dump assembly. Will not go beyond rotation sensors.
//Added for testing phase, can be removed when no longer needed.
//In order to actuate rotation motor, the user needs to be hold only
//L1,L2,R1,Square, and X and then move the right vertical joystick axis
//Up is positive rotation, down is negative rotation.
if( (PS2CMD2 == ((1<<PS2_L1)|(1<<PS2_L2)|(1<<PS2_R1)|...
    (1<<PS2_SQUARE)|(1<<PS2_X))) && (PS2CMD1 == 0x00) )
      {
      RotationSpeed = LeftVertical;
      }
if( RotationSpeed < 0 )                         //if positive rotation
      {
      if( !(PIND & (1<<SENSOR_ROTATION_180)) )//180 sensor is tripped
            RotationSpeed = 0;                //zero rotation speed
      }
if( RotationSpeed > 0 )                         //if negative rotation
      {
      if( !(PIND & (1<<SENSOR_ROTATION_0)) )  //and 0 sensor is tripped
            RotationSpeed = 0;                //zero rotation speed
      }

// END temporary section
//************************************************************

//Finished setting the vertical and rotation motor speeds
//***********************************************************************

//Set clam speeds
//First set ClamInvert
if(PS2CMD2 == ((1<<PS2_L2)|(1<<PS2_R2)|(1<<PS2_X))) //invert clam
            //joysticks if L2,R2, and X are pressed simultaneously
      {
      if(!ClamInvertFlag)           //if ClamInvertFlag is not set
            {
            ClamInvert = !ClamInvert; //invert ClamInvert
            }
      ClamInvertFlag = 1;           //signal an invert
      }
else
      ClamInvertFlag = 0;           //signal buttons released

if(ClamInvert)
      {
      LeftClamSpeed  = RightHorizontal;
      RightClamSpeed = LeftHorizontal;
      }
else
      {
      LeftClamSpeed  = -LeftHorizontal;
      RightClamSpeed = -RightHorizontal;
      }
//Finished setting clam invert



//L1 is the enable button on the PS2 controller for the clams
if( PS2CMD2 != (1<<PS2_L1) )
      {
      LeftClamSpeed = 0; //if button is not pressed, zero motor speeds
      RightClamSpeed = 0;
      }

//Check clam sensors
if( !(PINA & (1<<SENSOR_CLAM_LEFT)) )      //Is left sensor tripped?
      if(LeftClamSpeed>0)                  //Is clam tiring to open?
            LeftClamSpeed = 0;             //Stop clam from opening
if( !(PINA & (1<<SENSOR_CLAM_RIGHT)) )     //Is right sensor tripped?
      if(RightClamSpeed<0)                 //Is clam tiring to open?
            RightClamSpeed = 0;            //Stop clam from opening
//Finished setting clam speeds
//*****************************************************************
} // END main if statement

else        //PS2 controller is not in analog mode and something is
            //wrong, stop machine
```

```
{
VerticalSpeed = 0;
RotationSpeed = 0;
RightClamSpeed = 0;
LeftClamSpeed = 0;
PS2Init();              //attempt to reestablish communication
//Function to turn off AX3500
//Other necessary functions
}

//Send motor speeds and brakes
//**********************************************************************
//Send vertical motor speed
PULSEOut( VerticalSpeed, PULSE_PIN_VERTICAL );  //sends motor command
                                    //to AX3500. Later can be replaced
                                    //with RS232 serial command.
if(VerticalSpeed == 0)  //enable or disable brake
      {
      PORTA &= ~(1<<DRA_VERTICAL_BRAKE);  //if speed ==0, enable brake
                                    //by setting DRA_VERTICAL_BRAKE = 0
      if(LastVerticalBrake==0)      //if last time brake was disabled
            {
            _delay_ms(50);          //delay for debouncing brake
            LastVerticalBrake = 1;  //current brake setting becomes
            }                       //last brake setting
      }
else
      {
      PORTA |= (1<<DRA_VERTICAL_BRAKE);   //else disable brake by
                                    //setting DRA_VERTICAL_BRAKE = 1
      if(LastVerticalBrake==1)      //if last time brake was enabled
            {
            _delay_ms(50);
            LastVerticalBrake = 0;  //current brake setting becomes
            }                       //last brake setting
      }
//Finished sending vertical motor speed
//**********************************************************************
```

```
//Send rotation motor speed
PULSEOut( RotationSpeed, PULSE_PIN_ROTATION );

if(RotationSpeed == 0)
      {
      PORTA &= ~(1<<DRA_ROTATION_BRAKE);  //if speed ==0 then enable
                              //brake by setting DRA_ROTATION_BRAKE = 0
      if(LastRotationBrake==0)
            {
            _delay_ms(50);
            LastRotationBrake = 1;
            }
      }
else
      {
      PORTA |=  (1<<DRA_ROTATION_BRAKE);  //else disable brake by
                                    //setting DRA_ROTATION_BRAKE = 1
      if(LastRotationBrake==1)
            {
            _delay_ms(50);
            LastRotationBrake = 0;
            }
      }
//Finished sending rotation motor speed
//*********************************************************************

SetRightClam(RightClamSpeed);
SetLeftClam(LeftClamSpeed);
}

return 0;
} // END  Main Loop

// END Main Function

// END Main Code

//*********************************************************************
```

```
/*********************************************************
This is the main include file for the DRA control software (DRA.h)
Version 1.0
Last modified:    October 15th 2009
Created:          August 10th 2009
Code by Matthew H. Jones

Notes and Revisions:
*/
//*********************************************************
#ifndef __DRA_H__  //ensure DRA.h is not included in the code twice
#define __DRA_H__

//other necessary includes
#include<avr/io.h> //includes all pin names and dedicated registers
#define F_CPU 20000000  //needed for delay.h
#include<util/delay.h>  //includes _delay_ms(int time) and
                        //_delay_us(int time)
#include<LCD.h>               //includes all LCD control funcitons
#include<SPI.h>               //includes all SPI functions necessary to
                        //communicate with second microcontroller,
                        //like setMotor1(int speed)
#include<PS2controller.h> //includes all commands to initialize and
                          //read PS2 controller
#include<USART.h>         //includes commands to use USART to
                      //communicate via RS232 (not complete or working)
#include<PULSE.h>       //includes commands for creating RC pulses to
                        //send to AX3500

#include<LCD.c>         //C code for LCD functions (must be included or
                        //you will get a build error)
#include<SPI.c>         //C code for SPI functions (must be included or
                        //you will get a build error)

//Rename some stock functions for readability
#define SetRightClam(speed)               setMotor1(speed)
#define BrakeRightClam(void)        brakeLowMotor1(void)
#define SetLeftClam(speed)                setMotor2(speed)
```

```
#define BrakeLeftClam(void)                brakeLowMotor2(void)

//Define PINS for DRA sensors
//Pins PA0 - PA3 are reserved for PS2 communication
//Pins PD0 and PD1 are reserved for USART and RC pulse generation
#define SENSOR_ROTATION_0      PD2   // 0 degree rotation sensor
#define SENSOR_ROTATION_180    PD3   // 180 degree rotation sensor
#define SENSOR_VERTICAL_TOP    PD4   // Top vertical track sensor
#define SENSOR_VERTICAL_BOTTOM PD5   // Bottom vertical track sensor
#define SENSOR_VERTICAL_DOCKED PD6   // Docked vertical track sensor
#define SENSOR_CLAM_LEFT       PA6   // Left clam vain sensor
#define SENSOR_CLAM_RIGHT      PA7   // Right clam vain sensor

//Define pins for Brakes
#define DRA_VERTICAL_BRAKE     PA4
#define DRA_ROTATION_BRAKE     PA5



#define SENSOR_SHUTTLE_SENSORS
      (1<<SENSOR_ROTATION_0)|(1<<SENSOR_ROTATION_180)|...
      (1<<SENSOR_VERTICAL_TOP)|(1<<SENSOR_VERTICAL_BOTTOM)|...
      (1<<SENSOR_VERTICAL_DOCKED)

//Define cases for SWITCH based on sensor configuration
#define DRA_TOP_0 (1<<SENSOR_VERTICAL_TOP)|(1<<SENSOR_ROTATION_0)
      //Shuttle at top with 0 degree rotation
#define DRA_TOP_180 (1<<SENSOR_VERTICAL_TOP)|(1<<SENSOR_ROTATION_180)
      //Shuttle at top with 180 degree rotation
#define DRA_DOCKED (1<<SENSOR_VERTICAL_DOCKED)|(1<<SENSOR_ROTATION_180)
      //Shuttle docked
#define DRA_BOTTOM (1<<SENSOR_VERTICAL_BOTTOM)|(1<<SENSOR_ROTATION_0)
      //Shuttle at bottom of track (fully deployed)
#define DRA_MID_0 (1<<SENSOR_ROTATION_0)
      //Shuttle in the middle of track at 0 degrees
#define DRA_MID_180 (1<<SENSOR_ROTATION_180)
      //Shuttle in the middle of track at 180 degrees
#define DRA_TOP_MID (1<<SENSOR_VERTICAL_TOP)
      //Shuttle at the top of track in mid rotation
```

```
//Function Prototypes
void DRAInit(void);
void Top0(void);
void Top180(void);
void Docked(void);
void Bottom(void);
void Mid0(void);
void Mid180(void);
void TopMid(void);
void Error(void);

//variables
int LeftClamSpeed;              //Positive is open, negative is closed
int RightClamSpeed;             //Positive is open, negative is closed
int VerticalSpeed;              //Positive is up, negative is down
int RotationSpeed;              //Positive is rotate towards dump,
                                //negative is rotate towards deploy

//Functions
//*******************************************************************
//BEGIN DRAInit()
//Purpose: to initialize the remaining pins of the DRA for the sensors
//and brakes
//Input: NONE
//Output: NONE
//*******************************************************************
void DRAInit(void)
{
//Initialize output pins for brakes
DDRA |= (1<<DRA_ROTATION_BRAKE)|(1<<DRA_VERTICAL_BRAKE);
                  //Set pins as outputs
PORTA &= ~((1<<DRA_ROTATION_BRAKE)|(1<<DRA_VERTICAL_BRAKE));
                  //Make sure brakes are actuated (brake active low)



//Make sure sensor ports are inputs
```

```
DDRA &= ~(1<<SENSOR_CLAM_LEFT)|(1<<SENSOR_CLAM_RIGHT);
DDRD &= ~(1<<SENSOR_ROTATION_0)|(1<<SENSOR_ROTATION_180)|...
         (1<<SENSOR_VERTICAL_TOP)|(1<<SENSOR_VERTICAL_BOTTOM)|...
         (1<<SENSOR_VERTICAL_DOCKED);

//Setup pull up resistors for sensors
PORTA |= (1<<SENSOR_CLAM_LEFT)|(1<<SENSOR_CLAM_RIGHT);
PORTD |= (1<<SENSOR_ROTATION_0)|(1<<SENSOR_ROTATION_180)|...
         (1<<SENSOR_VERTICAL_TOP);

//Any add-on initializations
} // END DRAInit()

//*********************************************************************//BEGI
N Top0()
//Purpose: Calls the PS2Get function and restricts and allows motor
//operations based on shuttle being at top and 0 degree rotation
//Does not issue motor commands
//Input: NONE
//Output: NONE
//*********************************************************************
void Top0(void)
{
if(RightVertical > 0)
      {
      VerticalSpeed = 0;
      RotationSpeed = RightVertical;
      }
else if(RightVertical < 0)
      {
      VerticalSpeed = RightVertical;
      RotationSpeed = 0;
      }
else
      {
      VerticalSpeed = 0;
      RotationSpeed = 0;
      }
```

```
} // END Top0()

//*********************************************************************
//BEGIN Top180()
//Purpose: Calls the PS2Get function and restricts and allows motor
//operations based on shuttle being at top and 180 degrees rotation
//Does not issue motor commands
//Input: NONE
//Output: NONE
//*********************************************************************
void Top180(void)
{
if(RightVertical > 0)
      {
      VerticalSpeed = -RightVertical;
      RotationSpeed = 0;
      }

else if(RightVertical<0)
      {
      VerticalSpeed = 0;
      RotationSpeed = RightVertical;
      }
else
      {
      VerticalSpeed = 0;
      RotationSpeed = 0;
      }
} // END Top180()

//*********************************************************************
//BEGIN Docked()
//Purpose: Calls the PS2Get function and restricts and allows motor \
//operations based on shuttle being at the docked position
//Does not issue motor commands
//Input: NONE
//Output: NONE
//*********************************************************************
```

```
void Docked(void)
{
if(RightVertical > 0)
	{
	VerticalSpeed = 0;
	RotationSpeed = 0;
	}
else if(RightVertical<0)
	{
	VerticalSpeed = RightVertical;
	RotationSpeed = 0;
	}
else
	{
	RotationSpeed = 0;
	VerticalSpeed = 0;
	}
} // END Docked()

//*********************************************************************
//BEGIN Bottom()
//Purpose: Calls the PS2Get function and restricts and allows motor
//operations based on shuttle being at the bottom (fully deployed)
//Does not issue motor commands
//Input: NONE
//Output: NONE
//*********************************************************************
void Bottom(void)
{
RotationSpeed = 0;
if(RightVertical > 0)
	{
	VerticalSpeed = RightVertical;
	}




else
```

```
      {
      VerticalSpeed = 0;
      }
} // END Bottom()

//*********************************************************************
//BEGIN Mid0()
//Purpose: Calls the PS2Get function and restricts and allows motor
//operations based on shuttle being at mid track and 0 degree rotation
//Does not issue motor commands
//Input: NONE
//Output: NONE
//*********************************************************************void
Mid0(void)
{
RotationSpeed = 0;
VerticalSpeed = RightVertical;
if(VerticalSpeed < -80)  //to limit downward speed
      VerticalSpeed = -80;
} // END Mid0()

//*********************************************************************
//BEGIN Mid180()
//Purpose: Calls the PS2Get function and restricts and allows motor
//operations based on shuttle being at mid track and 180 degrees
//rotation. Does not issue motor commands
//Input: NONE
//Output: NONE
//*********************************************************************
void Mid180(void)
{
RotationSpeed = 0;
VerticalSpeed = -RightVertical;
if(VerticalSpeed < -80)  //figure out direction
      VerticalSpeed = -80;
} // END Mid180()

//*********************************************************************
```

```
//BEGIN TopMid()
//Purpose: Calls the PS2Get function and restricts and allows motor
//operations based on shuttle being at top in the middle of rotation
//Does not issue motor commands
//Input: NONE
//Output: NONE
//*********************************************************************
void TopMid(void)
{
VerticalSpeed = 0;
RotationSpeed = RightVertical;
} // END TopMid()







//*********************************************************************
//BEGIN Error()
//Purpose: Calls the PS2Get function (to maintain connectivity)
//Additional functions should be added as development continues
//Input: NONE
//Output: TBD
//*********************************************************************
void Error(void)
{
//DRA_Error = 1;  //set error flag
VerticalSpeed = 0;
RotationSpeed = 0;
//other error protocol
} // END Error()

#endif

/********************************************************
This is the control include file for reading the PS2 controller
(PS2Controller.h)
```

```
Version 3.0
Last modified:    October 20th 2009
Code by Matthew H. Jones
Notes and Revisions:

*/
//*********************************************************

#ifndef __PS2_H_
#define __PS2_H_

//Define pins
//PD0 and PD1 are reserved for USART
#define PS2_DATA  PA0   //data
#define PS2_CMD   PA1   //command
#define PS2_ATT   PA2   //attention
#define PS2_CLK   PA3   //clock

//Define constants for PS2 communication
#define PS2_START_CMD         0x01
#define PS2_REQUEST_DATA      0x42
#define PS2_CONFIG_MODE       0x43
#define PS2_CONFIG_MODE2      0x01
#define PS2_CONFIG_ANALOG     0x44
#define PS2_CONFIG_VIBRATION  0x4D
#define PS2_CONFIG_EXIT       0x00
#define PS2_SET_ANALOG        0x01
#define PS2_LOCK_ANALOG       0x03
#define PS2_SMALL_MOTOR       0x00
#define PS2_LARGE_MOTOR       0x01
#define PS2_DIGITAL_MODE      0x41
#define PS2_ANALOG_MODE       0x73
#define PS2_NO_CMD            0x00
//Define the locations of the different buttons in the data bytes
```

```
//Data byte 1 (PS2CMD1)
#define PS2_SELECT      0
#define PS2_L3          1
#define PS2_R3          2
#define PS2_START       3
#define PS2_UP          4
#define PS2_RIGHT       5
#define PS2_DOWN        6
#define PS2_LEFT        7
//Data byte 2 (PS2CMD2)
#define PS2_L2          0
#define PS2_R2          1
#define PS2_L1          2
#define PS2_R1          3
#define PS2_TRIANGLE    4
#define PS2_CIRCLE      5
#define PS2_X           6
#define PS2_SQUARE      7

//Define joystick deadbands
#define PS2_DEADBAND_1        70          //out of 256
#define PS2_RANGE_EXTEND_1    1.377       //determined empirically
#define PS2_DEADBAND_2        70
#define PS2_RANGE_EXTEND_2  1.377
#define PS2_DEADBAND_3        70
#define PS2_RANGE_EXTEND_3    1.377
#define PS2_DEADBAND_4        70
#define PS2_RANGE_EXTEND_4    1.377

//function prototypes
void PS2Init(void);
void PS2Display(void);
void PS2Get(int SmallMotor, int LargeMotor);
static unsigned char PS2ReadSend(unsigned char command);

//variables
      unsigned char PS2TEMP = 0x00;
      unsigned char PS2ID   = 0xFF;
      unsigned char PS2CMD1 = 0x00;
      unsigned char PS2CMD2 = 0x00;
      int RightHorizontal = 0x00;
      int RightVertical = 0x00;
      int LeftHorizontal = 0x00;
      int LeftVertical = 0x00;

//Function definitions

//*******************************************************************
//BEGIN PS2Init()
//Purpose: Initializes the pins for PS2 communication and sets the
//controller into analog mode and locks it
//Input: NONE
//Output: NONE
//*******************************************************************
void PS2Init(void)
{
      DDRA  |= (1<<PS2_CLK)|(1<<PS2_ATT)|(1<<PS2_CMD);
                                          //set pins as outputs
      PORTA |= (1<<PS2_CLK)|(1<<PS2_ATT)|(1<<PS2_DATA);
            //start clock, attention high, pull up resistor on DATA

while(PS2ID == 0xFF)    //loop until controller present
{
      PORTA &= ~(1<<PS2_ATT); //pull attention low
      PS2TEMP =   PS2ReadSend(PS2_START_CMD);
      PS2ID   =   PS2ReadSend(PS2_NO_CMD); //check for controller ID
      PORTA |= (1<<PS2_CLK);               //clock high
      PORTA |= (1<<PS2_ATT);               //attention high
}

//go into command mode
      PORTA &= ~(1<<PS2_ATT);             //pull attention low
      PS2TEMP =   PS2ReadSend(PS2_START_CMD);
      PS2TEMP =   PS2ReadSend(PS2_CONFIG_MODE);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_CONFIG_MODE2);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PORTA |= (1<<PS2_CLK);              //clock high
      PORTA |= (1<<PS2_ATT);              //attention high



//set controller to analog mode and lock
      PORTA &= ~(1<<PS2_ATT);             //pull attention low
      PS2TEMP =   PS2ReadSend(PS2_START_CMD);
      PS2TEMP =   PS2ReadSend(PS2_CONFIG_ANALOG);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_SET_ANALOG);
      PS2TEMP =   PS2ReadSend(PS2_LOCK_ANALOG);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PORTA |= (1<<PS2_CLK);              //clock high
      PORTA |= (1<<PS2_ATT);              //attention high

//enable vibration function
      PORTA &= ~(1<<PS2_ATT);             //pull attention low
      PS2TEMP =   PS2ReadSend(PS2_START_CMD);
      PS2TEMP =   PS2ReadSend(PS2_CONFIG_VIBRATION);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_SMALL_MOTOR);
      PS2TEMP =   PS2ReadSend(PS2_LARGE_MOTORs);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PORTA |= (1<<PS2_CLK);              //clock high
      PORTA |= (1<<PS2_ATT);              //attention high




//exit configuration mode
      PORTA &= ~(1<<PS2_ATT);             //pull attention low
      PS2TEMP =   PS2ReadSend(PS2_START_CMD);
      PS2TEMP =   PS2ReadSend(PS2_CONFIG_MODE);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2TEMP =   PS2ReadSend(0x5A);
      PS2TEMP =   PS2ReadSend(0x5A);
      PS2TEMP =   PS2ReadSend(0x5A);
      PS2TEMP =   PS2ReadSend(0x5A);
      PS2TEMP =   PS2ReadSend(0x5A);
      PORTA |= (1<<PS2_CLK);              //clock high
      PORTA |= (1<<PS2_ATT);              //attention high

} // END PS2Init()
//*********************************************************************
//BEGIN PS2Get()
//Purpose: Gets the PS2 controller configuration. Returns (as a global
//variable) the joystick positions as a value from -255 to +255
//Input: NONE
//Output: NONE
//*********************************************************************
void PS2Get(int SmallMotor, int LargeMotor)
{
      PORTA &= ~(1<<PS2_ATT);                   //pull attention low
      PS2TEMP =   PS2ReadSend(PS2_START_CMD);
      PS2ID   =   PS2ReadSend(PS2_REQUEST_DATA);
      PS2TEMP =   PS2ReadSend(PS2_NO_CMD);
      PS2CMD1 =   ~PS2ReadSend(SmallMotor);     //invert output so high
                                                //means button pressed
      PS2CMD2 =   ~PS2ReadSend(Large_Motor);    //invert output so high
                                                //means button pressed
      RightHorizontal =       PS2ReadSend(PS2_NO_CMD);
      RightVertical     =     PS2ReadSend(PS2_NO_CMD);
      LeftHorizontal  =       PS2ReadSend(PS2_NO_CMD);
      LeftVertical      =     PS2ReadSend(PS2_NO_CMD);
      PORTA |= (1<<PS2_CLK);                    //clock high
      PORTA |= (1<<PS2_ATT);                    //attention high

//Set the range of the joystick return values and apply the deadband,
//then extend range back to full
```

```
RightHorizontal = 2*RightHorizontal-256;  //positive right
if((RightHorizontal<=PS2_DEADBAND_1)&&...
   (RightHorizontal>=-PS2_DEADBAND_1))
      RightHorizontal = 0;

if(RightHorizontal > 0)       //extend range
 RightHorizontal = (RightHorizontal-PS2_DEADBAND_1)*PS2_RANGE_EXTEND_1;
if(RightHorizontal < 0)
 RightHorizontal = (RightHorizontal+PS2_DEADBAND_1)*PS2_RANGE_EXTEND_1;

RightVertical = -(2*RightVertical-256);
if((RightVertical<=PS2_DEADBAND_2)&&...
   (RightVertical>=-PS2_DEADBAND_2))
      RightVertical = 0;



if(RightVertical > 0)         //extend range
     RightVertical = (RightVertical-PS2_DEADBAND_2)*PS2_RANGE_EXTEND_2;
if(RightVertical < 0)
     RightVertical = (RightVertical+PS2_DEADBAND_2)*PS2_RANGE_EXTEND_2;

LeftHorizontal = 2*LeftHorizontal-256;          //positive right
if((LeftHorizontal<=PS2_DEADBAND_3)&&...
   (LeftHorizontal>=-PS2_DEADBAND_3))
      LeftHorizontal = 0;

if(LeftHorizontal > 0)        //extend range
    LeftHorizontal= (LeftHorizontal-PS2_DEADBAND_3)*PS2_RANGE_EXTEND_3;
if(LeftHorizontal < 0)
   LeftHorizontal = (LeftHorizontal+PS2_DEADBAND_3)*PS2_RANGE_EXTEND_3;

LeftVertical = (2*LeftVertical-256);
if((LeftVertical<=PS2_DEADBAND_4)&&(LeftVertical>=-PS2_DEADBAND_4))
      LeftVertical= 0;

if(LeftVertical > 0)          //extend range
      LeftVertical= (LeftVertical-PS2_DEADBAND_4)*PS2_RANGE_EXTEND_4;
if(LeftVertical < 0)
```

```
      LeftVertical = (LeftVertical+PS2_DEADBAND_4)*PS2_RANGE_EXTEND_4;
} //END PS2Get()

//*****************************************************************
//BEGIN PS2ReadSend()
//Purpose: The bitwise send command and read data function. This
//function is only to be called by the functions PS2Init() and PS2Get()
//and should never be called directly from the DRA code
//Input: PS2 commands
//Output: PS2 data
//*****************************************************************
unsigned char PS2ReadSend(unsigned char command)
//clock must be high before calling
{
      unsigned char PS2Data = 0x00;
      for(int i=0;i<8;i++)
      {
            PORTA &= ~(1<<PS2_CLK);       //set clock low
            if(bit_is_set(command, i))    //is PS2 command bit i set?
                  PORTA |= (1<<PS2_CMD);  //if yes set PS2_CMD bit
            else
                  PORTA &= ~(1<<PS2_CMD); //if no clear PS2_CMD bit
            _delay_us(60);    //wait for PS2 controller to set data bit
            PORTA |= (1<<PS2_CLK);  //set clock high. Controller sets
                                    //and reads on clock change
            if(bit_is_set(PINA, PS2_DATA)) //is PS2_DATA bit set?
                  PS2Data |= (1<<i);      //if yes set PS2Data bit i
            else
                  PS2Data &= ~(1<<i);     //if no clear PS2Data bit i
            _delay_us(60);                                        //wait for
PS2 controller to
      }
      _delay_us(300);   //wait for PS2 controller to acknowledge
      return PS2Data;
} // END PS2ReadSend()

//*****************************************************************
//BEGIN PS2Display()
```

```
//Purpose: Displays the PS2 controller configuration on the LCD
//Only used for debugging
//Input: NONE
//Output: NONE
//**********************************************************************
void PS2Display(void)
{
      LCDClear();
      LCDHex(PS2ID);
      LCDMoveCursor(LCD_ROW_1);
      LCDBinary(~PS2CMD1);
      LCDAddString("  ");
      LCDBinary(~PS2CMD2);
      LCDMoveCursor(LCD_ROW_2);
      LCDInt(LeftVertical);
      LCDAddString("   ");
      LCDInt(RightVertical);
      LCDMoveCursor(LCD_ROW_3);
      LCDInt(LeftHorizontal);
      LCDAddString("   ");
      LCDInt(RightHorizontal);
} // END PS2Display()

#endif      //END __PS2_H_

/**********************************************************
This is the Pulse include (Pulse.h)
Sets up functions to deliver RC timed pulses
Version 1.0
Last modified:    September 23rd 2009
Created:          September 10th 2009
Code by Matthew H. Jones

Notes and Revisions:

*/
//**********************************************************
#ifndef __PULSE_H__
```

```
#define __PULSE_H__

//Define constants
#define PULSE_TIME_MIN        589   //found using oscilloscope
#define PULSE_TIME_MAX        1588  //found using oscilloscope
#define PULSE_TIME_OFFSET     1090  //found using oscilloscope

//Define pins
#define PULSE_PIN_VERTICAL    PD0
#define PULSE_PIN_ROTATION    PD1

//Function prototypes
void PULSEInit(void);
void PULSEOut(int RCPulse, int Motor);

//Function definitions
//**********************************************************************
//BEGIN PULSEInit()
//Purpose: Initialize PD0 and PD1 pins as outputs and sets them to zero
//Input: NONE
//Output: NONE
//**********************************************************************
void PULSEInit(void)
{
DDRD |= (1<<PULSE_PIN_VERTICAL)|(1<<PULSE_PIN_ROTATION);          //pins as
output
PORTD &= ~( (1<<PULSE_PIN_VERTICAL)|(1<<PULSE_PIN_ROTATION) );
      //initialize pins to zero
} // END PULSEInit()
//**********************************************************************
//BEGIN PULSEOut()
//Purpose: Sends the pulse commands for AX3500 motor controller
//Values determined by oscilloscope
//Input: Pulse width in microseconds and pin(s) to be pulsed
//Output: NONE
//**********************************************************************
void PULSEOut(int RCPulse, int MotorPin)
{
```

```
RCPulse = 2*RCPulse + 1090;   //offset
//make sure pulse width is within range
//1500us is achieved with a RCPulse input of 1090 (unsure of reason for
//discrepency)
if(RCPulse>PULSE_TIME_MAX)
      RCPulse = PULSE_TIME_MAX;
if(RCPulse<PULSE_TIME_MIN)
      RCPulse = PULSE_TIME_MIN;

PORTD |= (1<<MotorPin);       //output one on motor pin
_delay_us(RCPulse);           //delay for speed and direction 1ms - 2ms
                              //(1.5ms is stop)

PORTD &= ~(1<<MotorPin);      //output zero
} // END PULSEOut()

#endif

/***********************************************************
This is the USART include (USART.h)
Sets up functions send and receive serial commands
9600 baud, 7-bit data, 1 Start bit, 1 Stop bit, Even Parity
Version 1.0
Last modified:    August 28th 2009
Created:          August 5th 2009
Code by Matthew H. Jones

Notes and Revisions:

*/
//***********************************************************
#ifndef __USART_H
#define __USART_H

//define constants and global variables
#define BAUD_RATE       9600
#define USART_UBRR      129
unsigned char temp1, temp2;
```

```
//function prototypes
int USARTInit(void);
void USARTTransmit(unsigned char data);
unsigned char USARTReceive(void);




//functions
//*********************************************************************
//BEGIN USARTInit(void)
//Purpose: Initializes serial communication
//Input: NONE, Output: NONE
//*********************************************************************
int USARTInit(void)
{
UBRR0H = (unsigned char)(USART_UBRR>>8);  //set baud rate
      //there is an error in the AVR644 data sheet example code
UBRR0L = (unsigned char) USART_UBRR;
      //it uses registers UBRRH0 and UBRRL0, which is wrong
//enable receiver and transmitter
UCSR0B = (1<<RXEN0)|(1<<TXEN0);

//set frame format: 7 data, 1 start, 1 stop, even parity
UCSR0C = (1<<UPM01)|(2<<UCSZ00); //(even parity) (7 data bits)
                         //default 1 start, 1 stop
return 0;
} // END USARTInit

//*********************************************************************
//BEGIN USARTTransmit(unsigned char data)
//Purpose: Transmits serial data
//Input: serial data to be transmitted of type char
//Output: NONE
//*********************************************************************
void USARTTransmit(unsigned char data)
{
//wait for transmit buffer to be ready
while( !( UCSR0A & (1<<UDRE0)) );   // do nothing while waiting

//put data into the buffer
UDR0 = data;

} // END USARTTransmit

//*********************************************************************
//BEGIN USARTReceive(void)
//Purpose: Receives serial data
//Input: NONE
//Output: Serial data from receive pin
//*********************************************************************
unsigned char USARTReceive(void)
{

//wait for data to be received
while( !( UCSR0A & (1<<RXC0)) );    //do nothing while waiting

return UDR0;

} // END USARTReceive

#endif // __USART_H
```

APPENDIX C: COMPLETE PS2 PACKET SEQUENCES

	Byte #	Line	Discription
Header	1	CMD	0x01: Initiate information transfer
		DATA	0xFF: Non-functional
	2	CMD	0x43: Enter/Exit configuration mode
		DATA	0x41: PS2 status ID byte
	3	CMD	0x00: Non-functional
		DATA	0x5A: Non-functional
Configuration Parameters	4	CMD	0x01: Enter configuration mode
		DATA	PS2 button configuration byte 1
	5	CMD	0x00: No command
		DATA	PS2 button configuration byte 2
Header	6	CMD	0x01: Initiate information transfer
		DATA	0xFF: Non-functional
	7	CMD	0x44: Analog/Digital mode select
		DATA	0xF3: PS2 status ID byte
	8	CMD	0x00: Non-functional
		DATA	0x5A: Non-functional
Configuration Parameters	9	CMD	0x01: Analog mode select
		DATA	PS2 button configuration byte 1
	10	CMD	0x03: Analog mode lock
		DATA	PS2 button configuration byte 2
	11	CMD	0x00: No command
		DATA	PS2 right joystick horizontal
	12	CMD	0x00: No command
		DATA	PS2 right joystick vertical
	13	CMD	0x00: No command
		DATA	PS2 left joystick horizontal
	14	CMD	0x00: No command
		DATA	PS2 left joystick vertical
Header	15	CMD	0x01: Initiate information transfer
		DATA	0xFF: Non-functional
	16	CMD	0x43: Enter/Exit configuration mode
		DATA	0xF3: PS2 status ID byte
	17	CMD	0x00: Non-functional
		DATA	0x5A: Non-functional
Configuration Parameters	18	CMD	0x00: Exit configuration mode
		DATA	PS2 button configuration byte 1
	19	CMD	0x5A: Non-functional
		DATA	PS2 button configuration byte 2
	20	CMD	0x5A: Non-functional
		DATA	PS2 right joystick horizontal
	21	CMD	0x5A: Non-functional
		DATA	PS2 right joystick vertical
	22	CMD	0x5A: Non-functional
		DATA	PS2 left joystick horizontal
	23	CMD	0x5A: Non-functional
		DATA	PS2 left joystick vertical

Table C- 1: Command Sequence to Lock PS2 Controller in Analog Mode [32]

	Byte #	Line	Discription
Header	1	CMD	0x01: Initiate information transfer
		DATA	0xFF: Non-functional
	2	CMD	0x43: Enter/Exit configuration mode
		DATA	0x41: PS2 status ID byte
	3	CMD	0x00: Non-functional
		DATA	0x5A: Non-functional
Configuration Parameters	4	CMD	0x01: Enter configuration mode
		DATA	PS2 button configuration byte 1
	5	CMD	0x00: No command
		DATA	PS2 button configuration byte 2
Header	6	CMD	0x01: Initiate information transfer
		DATA	0xFF: Non-functional
	7	CMD	0x4D: Vibration Mode Enable
		DATA	0xF1: PS2 status ID byte
	8	CMD	0x00: Non-functional
		DATA	0x5A: Non-functional
Configuration Parameters	9	CMD	0x00: Map Byte 0 to Small Motor
		DATA	PS2 button configuration byte 1
	10	CMD	0x01: Map Byte 1 to Large Motor
		DATA	PS2 button configuration byte 2
Header	11	CMD	0x01: Initiate information transfer
		DATA	0xFF: Non-functional
	12	CMD	0x43: Enter/Exit configuration mode
		DATA	0xF1: PS2 status ID byte
	13	CMD	0x00: Non-functional
		DATA	0x5A: Non-functional
Configuration Parameters	14	CMD	0x00: Exit configuration mode
		DATA	PS2 button configuration byte 1
	15	CMD	0x5A: Non-functional
		DATA	PS2 button configuration byte 2

Table C- 2: Command Sequence to Enable PS2 Vibration Function [32]

Note: Table C.2 assumes the controller is in digital mode. If the controller is in analog mode, the first, second, and third PS2 status ID bytes will be 0x73, 0xF3, and 0xF3 respectively, and the configuration parameters will contain 6 bytes (2 bytes for the button configuration and 4 bytes for the analog joystick configurations). The command sequences of Tables C.1 and C.2 can be combined by leaving out bytes 15-23 of C.1 and bytes 1-5 of table C.2

	Byte #	Line	Discription
Header	1	CMD	0x01: Initiate information transfer
		DATA	0xFF: Non-functional
	2	CMD	0x42: Poll controller
		DATA	0x73: PS2 status ID byte
	3	CMD	0x00: Non-functional
		DATA	0x5A: Non-functional
Configuration Parameters	4	CMD	0xXX: Small Vibration Motor Command
		DATA	PS2 button configuration byte 1
	5	CMD	0xYY: Large Vibration Motor Command
		DATA	PS2 button configuration byte 2
	6	CMD	0x00: Non-functional
		DATA	PS2 right joystick horizontal
	7	CMD	0x00: Non-functional
		DATA	PS2 right joystick vertical
	8	CMD	0x00: Non-functional
		DATA	PS2 left joystick horizontal
	9	CMD	0x00: Non-functional
		DATA	PS2 left joystick vertical

Table C- 3: Command Sequence to Poll PS2 Controller in Analog Mode and Vibration Enabled [32]

Note: 0xXX is a hexadecimal value, with 0xFF turning on the small motor and all other values turning it off, and 0xYY is a proportional value, with 0x00 being full off and 0xFF being full on. Only the extreme values of YY were used.

APPENDIX D: ELECTRONICS AND MOTOR DATA SHEETS

CHERRY

DIGITAL VANE SENSOR

VN1015 Series

Magnetically activated digital vane sensor in a rugged, overmolded plastic housing with three pins or 3-wire flying leads.

Features

- Available in two operating temperature ranges
- Immune to moisture and dust
- Reliable and repeatable
- No mechanical contacts to wear out
- Operates from 4.5 to 24VDC
- Reverse battery protection to -24VDC
- Open collector (sinking or NPN) output can be used with bipolar or cmos logic circuits with suitable pull up resistor
- Sensor body material: glass-filled polyester
- Recommended vane parameters: low carbon material at least 0.040" thick, should penetrate to a depth <0.120" from bottom of sensor slot.

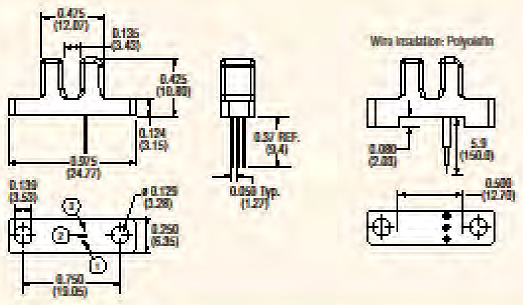
Specifications

Part Number	Operating Voltage Range (VDC)	Supply Current (mA max.)	Output	Output Saturation Voltage (mV max.)	Output Current (mA max.)	Operating Temp Range (°C)	Storage Temp Range (°C)	Termination
VN101501	4.5 – 24	6	3-pin sink	400	25	-40 to 85	-40 to 85	pins
VN101502	5.0 – 24	6	3-pin sink	400	25	-40 to 125	-40 to 125	pins
VN101503	4.5 – 24	6	3-wire sink	400	25	-40 to 85	-40 to 85	24 AWG x 150mm leads
VN101504	5.0 – 24	6	3-wire sink	400	25	-40 to 125	-40 to 125	24 AWG x 150mm leads

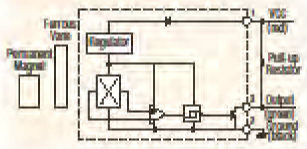
Notes: These sensors require the use of an external pull-up resistor, the value of which is dependent on the supply voltage. See page 18 for recommendations.
Pull-up resistor should be connected between output (Green) and Vcc (Red).

Dimensions inches (mm)

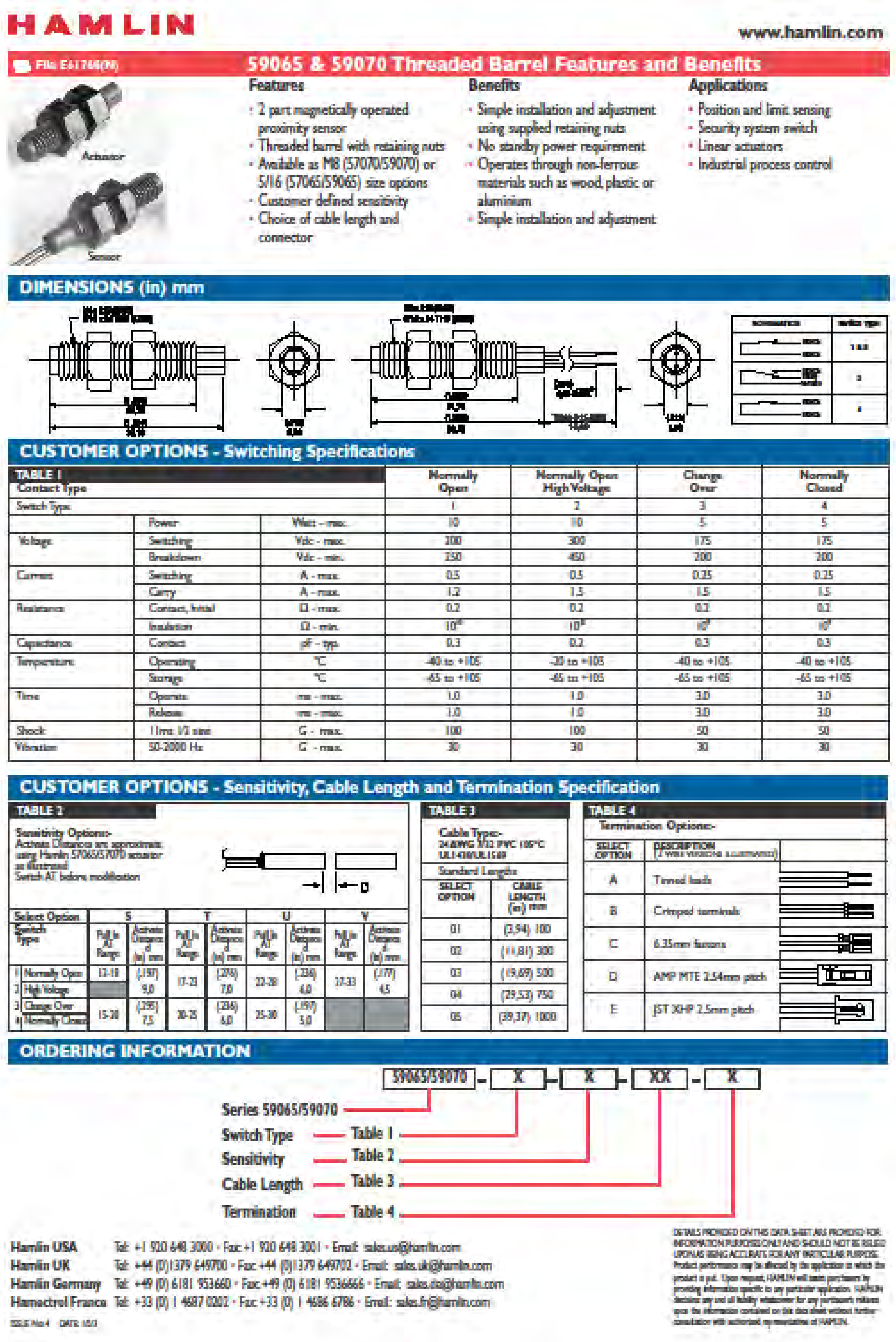
All tolerances ±0.005 (0.13) unless otherwise noted



Open Collector Sinking Block Diagram



Specifications subject to change without notice.



HAMLIN www.hamlin.com

File E61760(N)

59065 & 59070 Threaded Barrel Features and Benefits

Actuator
Sensor

Features

- 2 part magnetically operated proximity sensor
- Threaded barrel with retaining nuts
- Available as M8 (57070/59070) or 5/16 (57065/59065) size options
- Customer defined sensitivity
- Choice of cable length and connector

Benefits

- Simple installation and adjustment using supplied retaining nuts
- No standby power requirement
- Operates through non-ferrous materials such as wood, plastic or aluminium
- Simple installation and adjustment

Applications

- Position and limit sensing
- Security system switch
- Linear actuators
- Industrial process control

DIMENSIONS (in) mm

CUSTOMER OPTIONS - Switching Specifications

TABLE 1 Contact Type			Normally Open	Normally Open High Voltage	Change Over	Normally Closed
Switch Type			1	2	3	4
	Power	Watt - max.	10	10	5	5
Voltage	Switching	Vdc - max.	200	300	175	175
	Breakdown	Vdc - min.	250	450	200	200
Current	Switching	A - max.	0.5	0.5	0.25	0.25
	Carry	A - max.	1.2	1.5	1.5	1.5
Resistance	Contact, Initial	Ω - max.	0.2	0.2	0.2	0.2
	Insulation	Ω - min.	10^{10}	10^{8}	10^{9}	10^{9}
Capacitance	Contact	pF - typ.	0.3	0.2	0.3	0.3
Temperature	Operating	°C	-40 to +105	-20 to +105	-40 to +105	-40 to +105
	Storage	°C	-65 to +105	-65 to +105	-65 to +105	-65 to +105
Time	Operate	ms - max.	1.0	1.0	3.0	3.0
	Release	ms - max.	1.0	1.0	3.0	3.0
Shock	11ms 1/2 sine	G - max.	100	100	50	50
Vibration	50-2000 Hz	G - max.	30	30	30	30

CUSTOMER OPTIONS - Sensitivity, Cable Length and Termination Specification

TABLE 2

Sensitivity Options-
Actuate Distances are approximate using Hamlin 59065/57070 actuator as illustrated
Switch AT before modification

Select Option		S		T		U		V	
Switch Type		Pull-In AT Range	Actuate Distance d (in) mm	Pull-In AT Range	Actuate Distance d (in) mm	Pull-In AT Range	Actuate Distance d (in) mm	Pull-In AT Range	Actuate Distance d (in) mm
1	Normally Open	12-18	(.197) 9.0	17-23	(.276) 7.0	22-28	(.236) 6.0	27-33	(.177) 4.5
2	High Voltage	▨							
3	Change Over	15-20	(.295) 7.5	20-25	(.236) 6.0	25-30	(.197) 5.0	▨	▨
4	Normally Closed								

TABLE 3

Cable Type-
24 AWG 7/32 PVC 105°C
UL1430/UL1569

Standard Lengths

SELECT OPTION	CABLE LENGTH (in) mm
01	(3.94) 100
02	(11.81) 300
03	(19.69) 500
04	(29.53) 750
05	(39.37) 1000

TABLE 4

Termination Options-

SELECT OPTION	DESCRIPTION (2 wire versions illustrated)
A	Tinned leads
B	Crimped terminals
C	6.35mm fastons
D	AMP MTE 2.54mm pitch
E	JST XHP 2.5mm pitch

ORDERING INFORMATION

59065/59070 - X - X - XX - X

Series 59065/59070	
Switch Type	Table 1
Sensitivity	Table 2
Cable Length	Table 3
Termination	Table 4

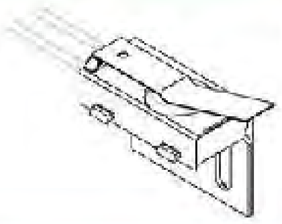
Hamlin USA	Tel: +1 920 648 3000 • Fax +1 920 648 3001 • Email: sales.us@hamlin.com
Hamlin UK	Tel: +44 (0)1379 649700 • Fax +44 (0)1379 649702 • Email: sales.uk@hamlin.com
Hamlin Germany	Tel: +49 (0) 6181 953660 • Fax +49 (0) 6181 9536666 • Email: sales.de@hamlin.com
Hamectrol France	Tel: +33 (0) 1 4687 0202 • Fax +33 (0) 1 4686 6786 • Email: sales.fr@hamlin.com

ISSUE No 4 DATE: 1/03

DETAILS PROVIDED ON THIS DATA SHEET ARE PROVIDED FOR INFORMATION PURPOSES ONLY AND SHOULD NOT BE RELIED UPON AS BEING ACCURATE FOR ANY PARTICULAR PURPOSE. Product performance may be affected by the application to which the product is put. Upon request, HAMLIN will assist purchasers by providing information specific to any particular application. HAMLIN disclaims any and all liability whatsoever for any purchaser's reliance upon the information contained on this data sheet without further consultation with authorised representatives of HAMLIN.

Control Products, Inc
280 Ridgedale Ave., E. Hanover, NJ 07936
Telephone: (973) 887-9400 Fax: (973) 887-5083
E-Mail: sales@cpi-nj.com Web site: www.cpi-nj.com

Item # E1101, E-Series Limit Brackets



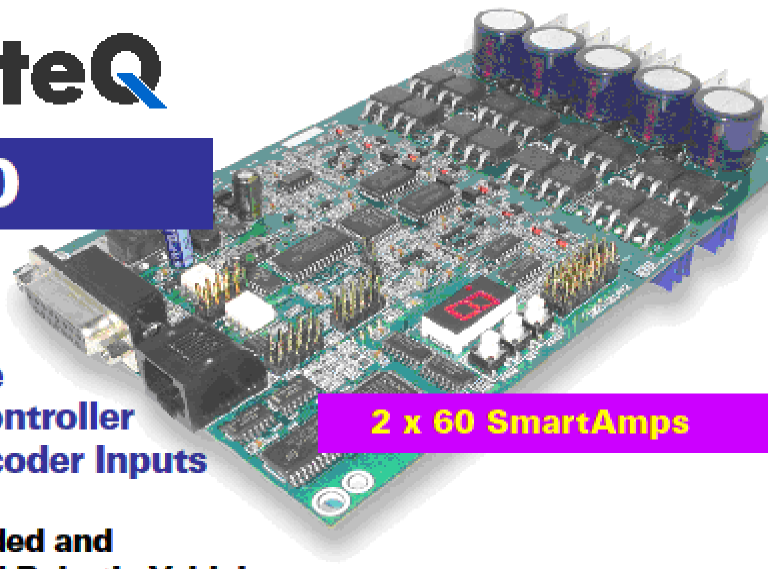
E-Series Limit Brackets
Opposite angle of E1100. Slotted mounting holes for 1" height adjustment.

SPECIFICATIONS

Method of Actuation	Lever
Mounting Type	Angle
Bracket Material	Stainless Steel
Compatible Switches	Any low button switch

RoboteQ

AX3500



Dual Channel Forward/Reverse Digital Robot Controller with Optical Encoder Inputs

2 x 60 SmartAmps

for Computer Guided and Remote Controlled Robotic Vehicles

Roboteq's AX3500 controller is a product family designed to convert commands received from a R/C radio, Analog Joystick, wireless modem, or microcomputer into high voltage and high current output for driving one or two DC motors. Designed for maximal ease-of-use in OEM applications, it is delivered with all necessary cables and hardware and is ready to use in minutes.

The controller's two channels can either be operated independently or mixed to set the direction and rotation of a vehicle by coordinating the motion on each side of the vehicle. The motors may be operated in open or closed loop speed mode. Using low-cost position sensors, they may also be set to operate as heavy-duty position servos.

The AX3500 may be ordered in one of many assembly options, depending whether optical encoder input is needed and depending on the application's power, number of channels, and thermal requirements.

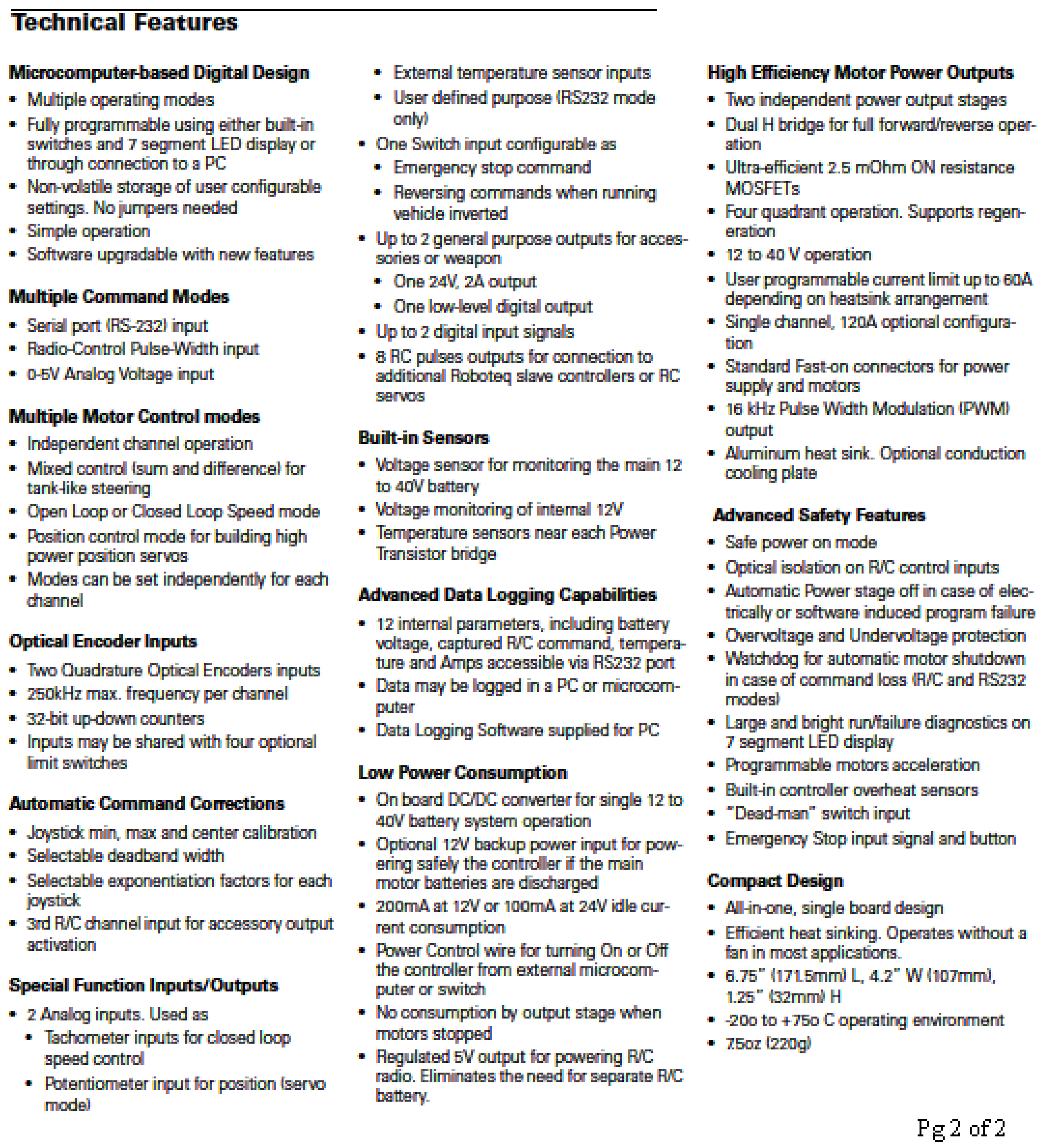
The AX2550 can be reprogrammed in the field with the latest features by downloading new operating software from Roboteq's web site. Numerous safety features are incorporated into the controller to ensure reliable and safe operation in the most demanding mobile robotic vehicle applications.

Applications

- Terrestrial and Underwater Robotic Vehicles
- Automatic Guided Vehicles
- Police and Military Robots
- Hazardous Material Handling Robots
- Telepresence Systems
- Animatronics
- Industrial Controls

Key Features	Benefits
Dual MCU digital design	Accurate, reliable, and fully programmable operation. Advanced algorithms
R/C mode support	Connects directly to simple, low cost R/C radios
RS232 Serial mode support	Connects directly to computers for autonomous operation or to wireless modem for two-way remote control
Analog mode support	Connects directly to analog joystick
Optical encoder inputs	Stable speed regardless of load. Accurate measurement of travelled distance
Built-in power drivers for two motors	Supports all common robot drive methods
60A output per channel	Gives robot strongest lifting or pushing power
Programmable current limitation	Protects controller, motors, wiring and battery.
Open loop or closed loop speed control	Low cost or higher accuracy speed control
Closed loop position control	Create low cost, ultra-high torque jumbo servos
Data Logging Output	Capture operating parameters in PC or PDA for analysis
Built-in DC/DC converter	Operates from a single 12V-40V battery
Field upgradable software	Never obsolete. Add features via the internet

Pg 1 of 2



Technical Features

Microcomputer-based Digital Design
- Multiple operating modes
- Fully programmable using either built-in switches and 7 segment LED display or through connection to a PC
- Non-volatile storage of user configurable settings. No jumpers needed
- Simple operation
- Software upgradable with new features

Multiple Command Modes
- Serial port (RS-232) input
- Radio-Control Pulse-Width input
- 0-5V Analog Voltage input

Multiple Motor Control modes
- Independent channel operation
- Mixed control (sum and difference) for tank-like steering
- Open Loop or Closed Loop Speed mode
- Position control mode for building high power position servos
- Modes can be set independently for each channel

Optical Encoder Inputs
- Two Quadrature Optical Encoders inputs
- 250kHz max. frequency per channel
- 32-bit up-down counters
- Inputs may be shared with four optional limit switches

Automatic Command Corrections
- Joystick min, max and center calibration
- Selectable deadband width
- Selectable exponentiation factors for each joystick
- 3rd R/C channel input for accessory output activation

Special Function Inputs/Outputs
- 2 Analog inputs. Used as
  - Tachometer inputs for closed loop speed control
  - Potentiometer input for position (servo mode)
  - External temperature sensor inputs
  - User defined purpose (RS232 mode only)
- One Switch input configurable as
  - Emergency stop command
  - Reversing commands when running vehicle inverted
- Up to 2 general purpose outputs for accessories or weapon
  - One 24V, 2A output
  - One low-level digital output
- Up to 2 digital input signals
- 8 RC pulses outputs for connection to additional Roboteq slave controllers or RC servos

Built-in Sensors
- Voltage sensor for monitoring the main 12 to 40V battery
- Voltage monitoring of internal 12V
- Temperature sensors near each Power Transistor bridge

Advanced Data Logging Capabilities
- 12 internal parameters, including battery voltage, captured R/C command, temperature and Amps accessible via RS232 port
- Data may be logged in a PC or microcomputer
- Data Logging Software supplied for PC

Low Power Consumption
- On board DC/DC converter for single 12 to 40V battery system operation
- Optional 12V backup power input for powering safely the controller if the main motor batteries are discharged
- 200mA at 12V or 100mA at 24V idle current consumption
- Power Control wire for turning On or Off the controller from external microcomputer or switch
- No consumption by output stage when motors stopped
- Regulated 5V output for powering R/C radio. Eliminates the need for separate R/C battery.

High Efficiency Motor Power Outputs
- Two independent power output stages
- Dual H bridge for full forward/reverse operation
- Ultra-efficient 2.5 mOhm ON resistance MOSFETs
- Four quadrant operation. Supports regeneration
- 12 to 40 V operation
- User programmable current limit up to 60A depending on heatsink arrangement
- Single channel, 120A optional configuration
- Standard Fast-on connectors for power supply and motors
- 16 kHz Pulse Width Modulation (PWM) output
- Aluminum heat sink. Optional conduction cooling plate

Advanced Safety Features
- Safe power on mode
- Optical isolation on R/C control inputs
- Automatic Power stage off in case of electrically or software induced program failure
- Overvoltage and Undervoltage protection
- Watchdog for automatic motor shutdown in case of command loss (R/C and RS232 modes)
- Large and bright run/failure diagnostics on 7 segment LED display
- Programmable motors acceleration
- Built-in controller overheat sensors
- "Dead-man" switch input
- Emergency Stop input signal and button

Compact Design
- All-in-one, single board design
- Efficient heat sinking. Operates without a fan in most applications.
- 6.75" (171.5mm) L, 4.2" W (107mm), 1.25" (32mm) H
- -20o to +75o C operating environment
- 7.5oz (220g)

Pg 2 of 2

Note: The complete user manual for the AX3500 can be downloaded from www.roboteq.com

Pololu

Orangutan X2 Robot Controller

Quick-Start Guide

Introduction

The Orangutan X2 is the third release in Pololu's line of Orangutan robot controllers. Like the original Orangutan and subsequent Baby Orangutan, the Orangutan X2 is designed to be a compact, high-performance control center for robotics and automation projects. The Orangutan X2's two-board design allows the unit to maintain the compactness characteristic of the Orangutan line while offering substantially more electrical and computational power: the X2 can deliver up to a horsepower across two motor channels, and the twin-microcontroller architecture allows maximum access to the primary microcontroller, an Atmel ATmega644 with 64KB of program space and 4KB of RAM. A battery, motors, and sensors can be connected directly to the module for quick creation of advanced robots.

Contacting Pololu

You can check the Pololu web site at `http://www.pololu.com/` for additional information about the Orangutan robot controllers, including color pictures, application examples, and troubleshooting tips.

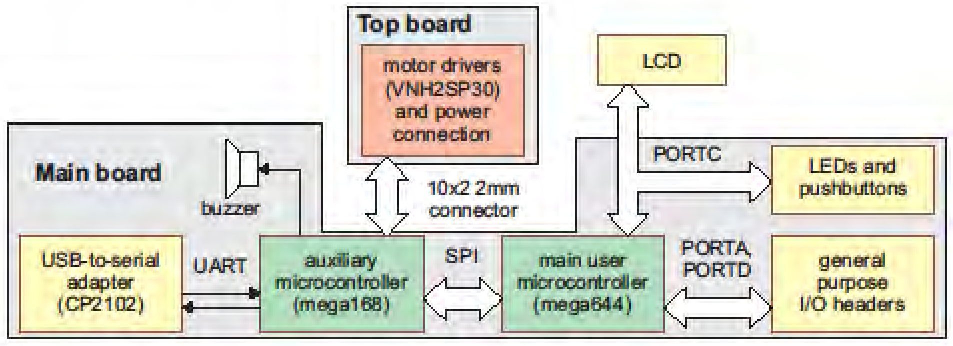
We would be delighted to hear from you about your project and about your experience with the Orangutan X2. You can contact us through our online feedback form or by email at `support@pololu.com`. Tell us what we did well, what we could improve, what you would like to see in the future, or anything else you would like to say!

Hardware Overview

Two circuit boards. A block diagram of the Orangutan X2 is shown below. The Orangutan X2 consists of two printed circuit boards connected by a 20-pin connector. The top board holds the high-power motor drivers and power terminals; the rest of the electronics, including the microcontrollers, is on the bottom board. The connections on the top board are symmetric, so until the connectors are soldered on, the board can be mounted in either orientation. The Orangutan is available with two motor driver options: the VNH3SP30 costs less, but has slightly lower performance; the VNH2SP30 can deliver more current and adds current sensing. Battery and motor leads (or leads to your favorite connector style) can be soldered directly to the top board, or the supplied terminal blocks can be used for quick convenient motor or power supply changes.

Two microcontrollers. The Orangutan X2 has two microcontrollers: an Atmel ATmega644 for the main application, and an auxiliary ATmega168 that interfaces to most of the dedicated hardware on the X2 and serves as a programmer for the main processor. The two-microcontroller design simplifies multitasking by relieving the main processor of common tasks such as motor control and melody generation, and the approach also leaves the mega644 completely unencumbered, allowing the mega644 hardware, such as timers and interrupts, to be used for your higher-level design.

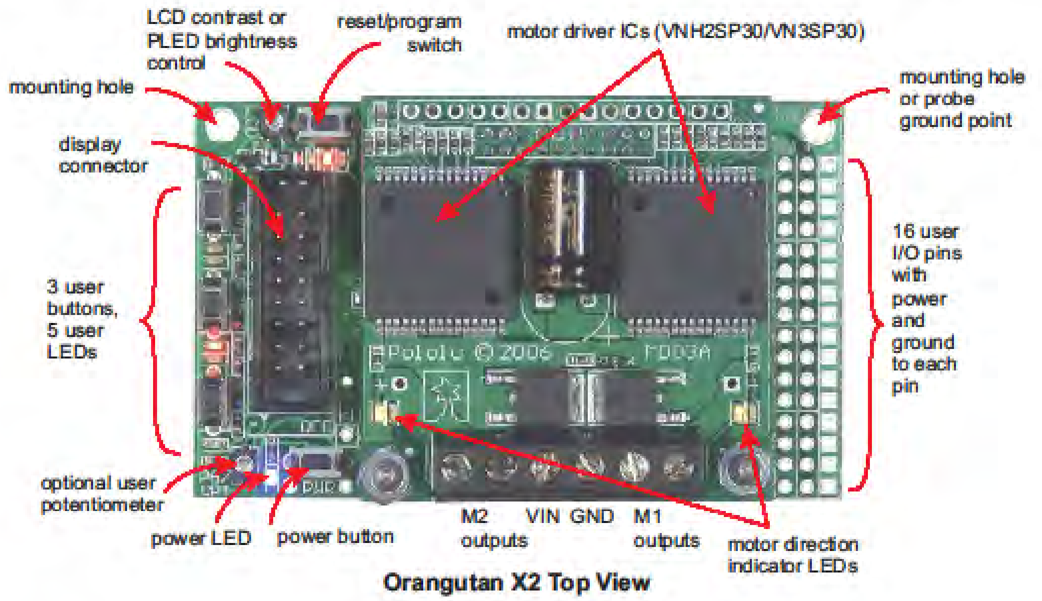
For more details, please check the complete schematic included at the end of this document.


Top board
motor drivers (VNH2SP30) and power connection
LCD
Main board
buzzer
10x2 2mm connector
PORTC
LEDs and pushbuttons
USB-to-serial adapter (CP2102)
UART
auxiliary microcontroller (mega168)
SPI
main user microcontroller (mega644)
PORTA, PORTD
general purpose I/O headers

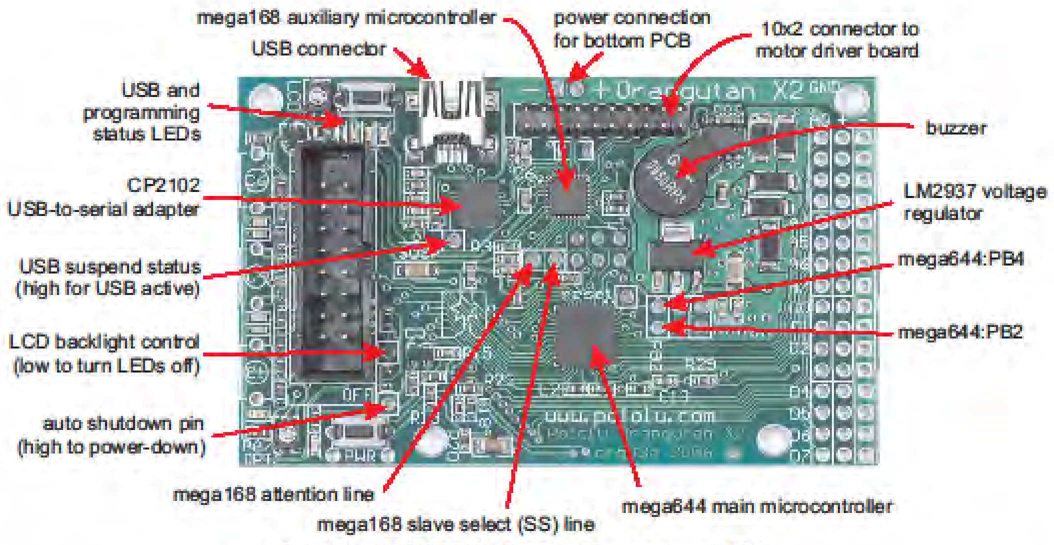
Orangutan X2 Block Diagram

Module Layout

The main features of the Orangutan X2 are indicated below. Most of the mega644 I/O lines come out to the 0.1" header along the right side, but the two uncommitted port B pins and the optional mega168 handshaking lines are in the middle of the board. The motor driver board has a few power supply capacitor options; the picture below shows a single capacitor bent over for a low-profile installation.


LCD contrast or PLED brightness control
reset/program switch
motor driver ICs (VNH2SP30/VN3SP30)
mounting hole
mounting hole or probe ground point
display connector
16 user I/O pins with power and ground to each pin
3 user buttons, 5 user LEDs
optional user potentiometer
power LED
power button
M2 outputs
VIN GND
M1 outputs
motor direction indicator LEDs

Orangutan X2 Top View

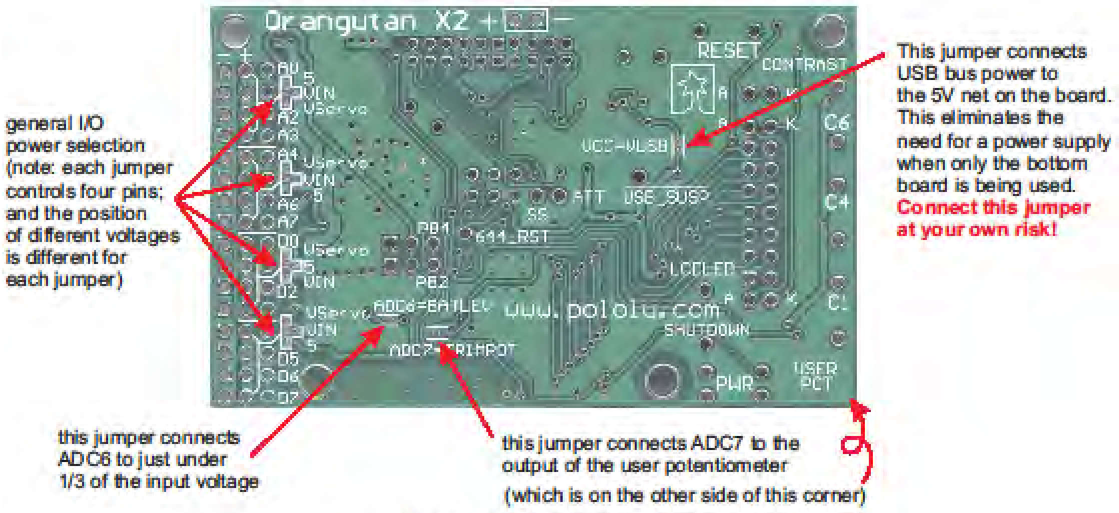

mega168 auxiliary microcontroller
USB connector
power connection for bottom PCB
10x2 connector to motor driver board
USB and programming status LEDs
buzzer
CP2102 USB-to-serial adapter
LM2937 voltage regulator
USB suspend status (high for USB active)
mega644:PB4
mega644:PB2
LCD backlight control (low to turn LEDs off)
auto shutdown pin (high to power-down)
mega168 attention line
mega168 slave select (SS) line
mega644 main microcontroller

Orangutan X2 Main Board, Component Side

page 2 of 6 org03a

Module Layout (continued)

Some hardware options on the Orangutan X2 are accessed by making or breaking solder bridges across surface-mount pads on the bottom side of the main printed circuit board; the pads are indicated below.


general I/O power selection (note: each jumper controls four pins; and the position of different voltages is different for each jumper)
This jumper connects USB bus power to the 5V net on the board. This eliminates the need for a power supply when only the bottom board is being used. Connect this jumper at your own risk!
this jumper connects ADC6 to just under 1/3 of the input voltage
this jumper connects ADC7 to the output of the user potentiometer (which is on the other side of this corner)
Orangutan X2
RESET
CONTRAST
VCC=VLSB
USE_SUSP
644_RST
LCDLED
ADC6=BATLEV
ADC7=TRIMPOT
www.pololu.com
SHUTDOWN
PWR
USER PCT

Orangutan X2 Bottom View

Power Connections and Limits

The Orangutan X2 power input is on the two lower, middle pins of the upper board. The operating range is **6-16V**. When using large motors, make sure the power supply and wiring can handle the current; you might also consider putting a fuse in line with the main power. The motor drivers are capable of delivering surges of up to 30 A, and continuous current will depend on the environment. The VNH3 is generally good for up to about 9 A continuous, and the VNH2 is good for up to about 14 A. Heat sinks can improve the motor driver performance.

The power to the main PCB is delivered through four pairs of pins on the 2mm connector, which limits the total power to the bottom board to approximately 6 A. The onboard 5 V regulator is good for up to 500 mA, but since the practical limit comes from power dissipation, the usable current will depend on the input voltage and the ambient temperature. The Vservo line is about 2V below the input voltage, and it can be used to power servos when the main supply is just a bit too high for servos, as with 6- or 7-cell NiMH battery packs. The limit for this supply is about 3A, but as with most power issues, it depends on how much heat the rest of the board is dissipating.

The Orangutan X2 is intended to be used as a single unit with both boards connected together. However, it may sometimes be convenient to work with just the lower board, without motors or a large power supply connected. In such cases, the two power input pins above the 20-pin connector can be used. In cases where very little power needs to be supplied outside the board, the USB port can also be used as a power source. **In this case, the power switch will not work, and your computer will be exposed to any voltage fluctuations on your Vcc line, so do so at your own risk.**

Power Button
The Orangutan X2 power is controlled by a pushbutton; push it to toggle the unit on and off. Because the power switch is operated by a pushbutton, many buttons can be used in parallel, allowing for external power buttons in cases where the main unit is difficult to access. Only power for the main board is switched; the motor driver board power is not switched.

The power consumption in the off state depends on the input voltage, but it is typically under 100 uA, most of which comes from the motor driver quiescent current and power supply capacitor leakage current. **Note: the power switch does not actually disconnect the power supply from the board, so even if the board is turned off, it is possible to do things like accidentally short-circuit the power supply!**

Connecting the Orangutan to a Computer

The mega168 microcontroller is the programmer for the main mega644 MCU. The mega168 performs this function by emulating an AVRISP programmer, which connects to a computer serial (COM) port and programs AVR microcontrollers via the SPI (serial peripheral interface) port. Instead of a standard serial port, the Orangutan X2 uses a USB-to-serial bridge that allows a USB connection to look like a COM port. Before connecting the Orangutan X2 to a computer, the driver must be installed to allow the computer's operating system to treat the USB connection as an old-fashioned serial connection. The driver and installation instructions are available on the Orangutan X2 web page.

Once the USB-to-serial driver is installed and the Orangutan X2 is connected, the mega168 can communicate with the computer through its serial port, and the green LED next to the USB connector will be lit. When programming the mega644, the Orangutan X2 will look like an AVRISP programmer; during normal operation, the mega168 can send and receive data to or from the computer (e.g. using a terminal program) for debugging or other purposes.

Programming the Orangutan X2

The Orangutan X2 can be programmed using any platform for which there is a USB driver and for which there is AVRISP-compatible programmer software. We recommend using Atmel's AVR Studio, an integrated development environment (IDE) that works with the free GCC C compiler and includes a simulator and other useful tools, including AVRISP support.

To enter programming mode, hold down the reset/programming button (next to the USB connector) for more than half a second. The buzzer will beep, and the yellow LED will turn on, indicating that you have entered programming mode. The mega168 will no longer respond to commands from the mega644, and it will wait for programming commands from the computer via the USB connection. When programming is in progress, the red LED will be lit. When programming completes, the mega644 is allowed to execute, but the mega168 will remain in programming mode until the reset button is pressed.

It is also possible to set the mega168 to always look out for programming commands. In that state, normal serial port use is unavailable, and any incoming serial data is treated as coming from the computer programming software. When programming is requested, the mega168 will program the mega644 and then reset itself and the mega644, allowing full operation to resume immediately upon completion of programming.

When programming the mega644, access to some fuse settings is not available. The most important setting is the clock source setting since the mega644 must be set for an external resonator, and the mega644 provides a 20 MHz clock to the mega168. In general, the fuses should only be changed rarely and with great care since the Orangutan X2 could become unresponsive.

Running the Orangutan X2

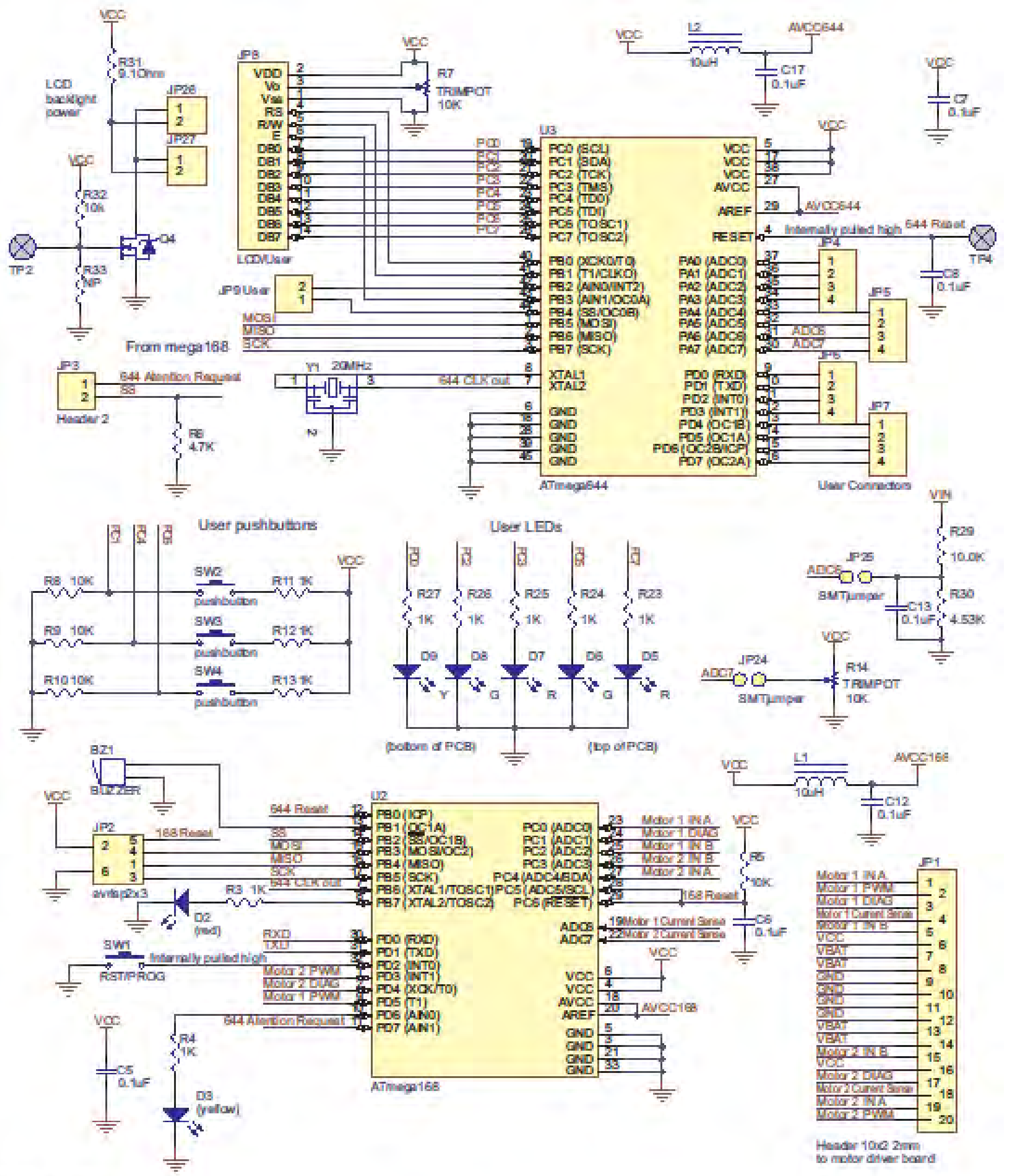
Using the Orangutan X2 is generally identical to using any other mega644-based project, and most of the mega644's resources are available to the user. The exceptions are the reset system and the SPI port, which are connected to the mega168.

Reset
Because the mega168 and mega644 need to stay synchronized, it is not desirable to reset the mega644 independently. The reset button does not connect directly to either processor's hardware reset line. Instead, the mega168 monitors the reset button and determines when to reset itself or the mega644. Typically, the mega168 will reset both processors, keep the mega644 reset while it initializes, and then finally allow the mega644 to begin execution. The reset button will not work during programming.

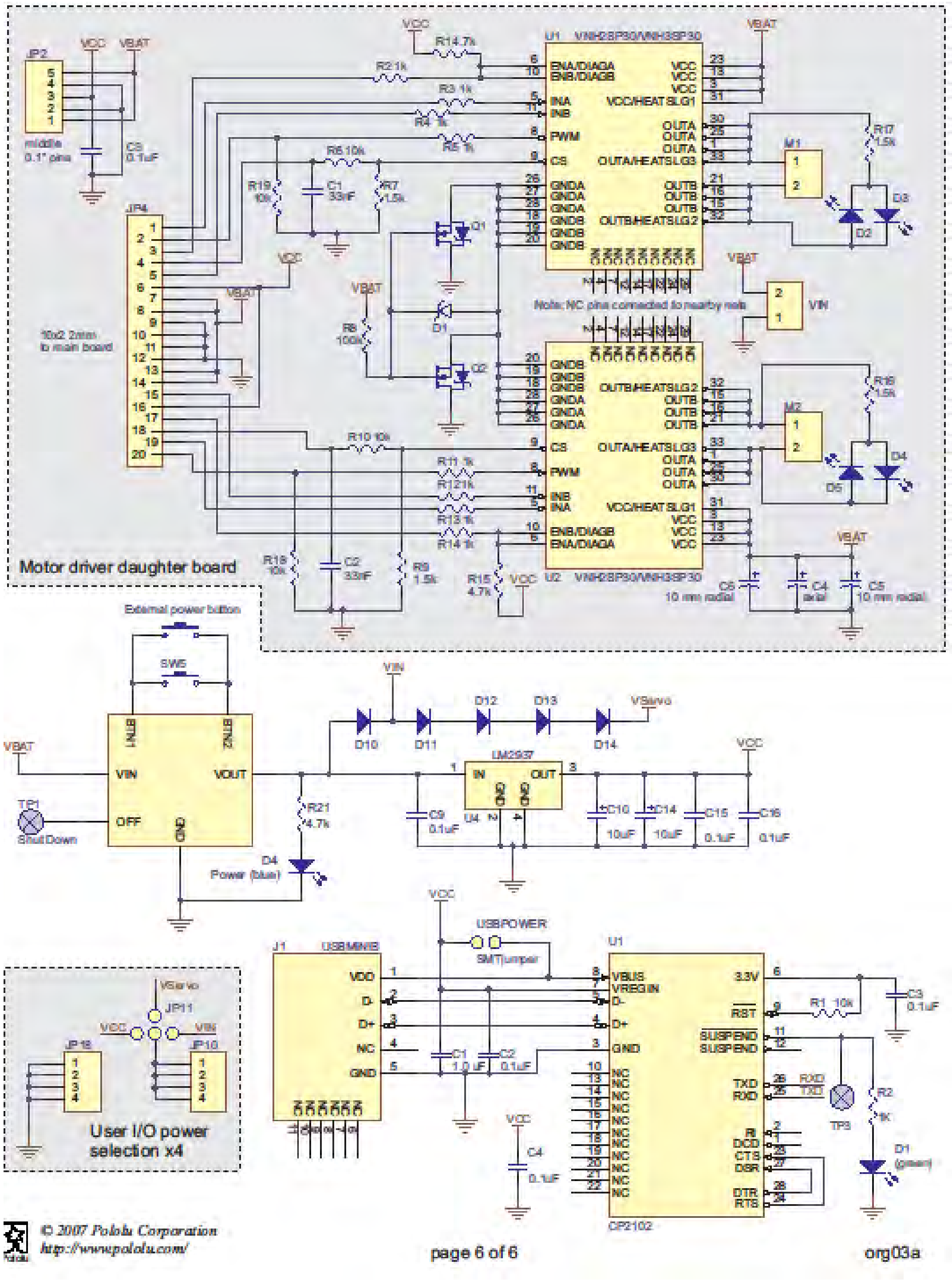
SPI Port
The SPI port is the main connection between the two MCUs. During programming, the mega168 becomes the master; during normal operation, the mega644 is the master and sends the mega168 commands via the SPI interface. The default setup of the Orangutan X2 assumes no other use of the SPI lines (the mega168's slave-select line is pulled down by a resistor). The SS line can instead be connected to one of the mega644 I/O lines, and the mega644 can then control multiple slave devices on the same SPI lines. It can also be desirable to use the SS line even without additional SPI devices since the SS line provides added robustness to the protocol.

The mega644 to mega168 SPI interface is detailed in a separate document; please see the Orangutan X2 web page for more details.

Orangutan X2 Robot Controller Schematic Diagram



© 2007 Pololu Corporation
http://www.pololu.com/
page 5 of 6
org03a



Note: Additional resources for the OX2 can be found at www.pololu.com

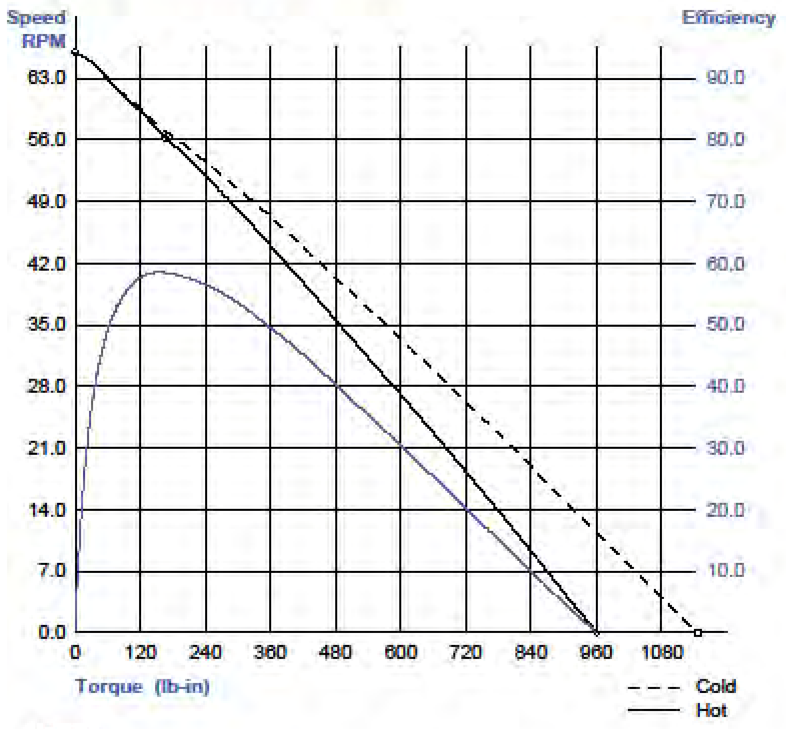
MOTORTEC™ DATA SHEET

Customer:	Groschopp.com	Quote No:	---
Application:	—	Date:	07/09/08
Prepared by:			

Motor:	PM 8018	
Voltage:	12 v DC	
Speed:	2250 rpm	2 Poles
Gearbox:	PL 7300	40:1 Ratio

Catalog Part No: **63505**

Speed, Efficiency Vs. Torque



Notes:

SPECIFICATIONS

Rating:

Speed:	56.3 rpm
Torque:	166.6lb-in
Current:	16.3 amps
Output:	111 watts
Output:	0.1487 HP

Duty Cycle:

On:	Continuous
Off:	

Efficiency:

Gearbox:	79.4 %
Motor:	73.7 %
System:	58.5 %

Start/Stall Conditions:

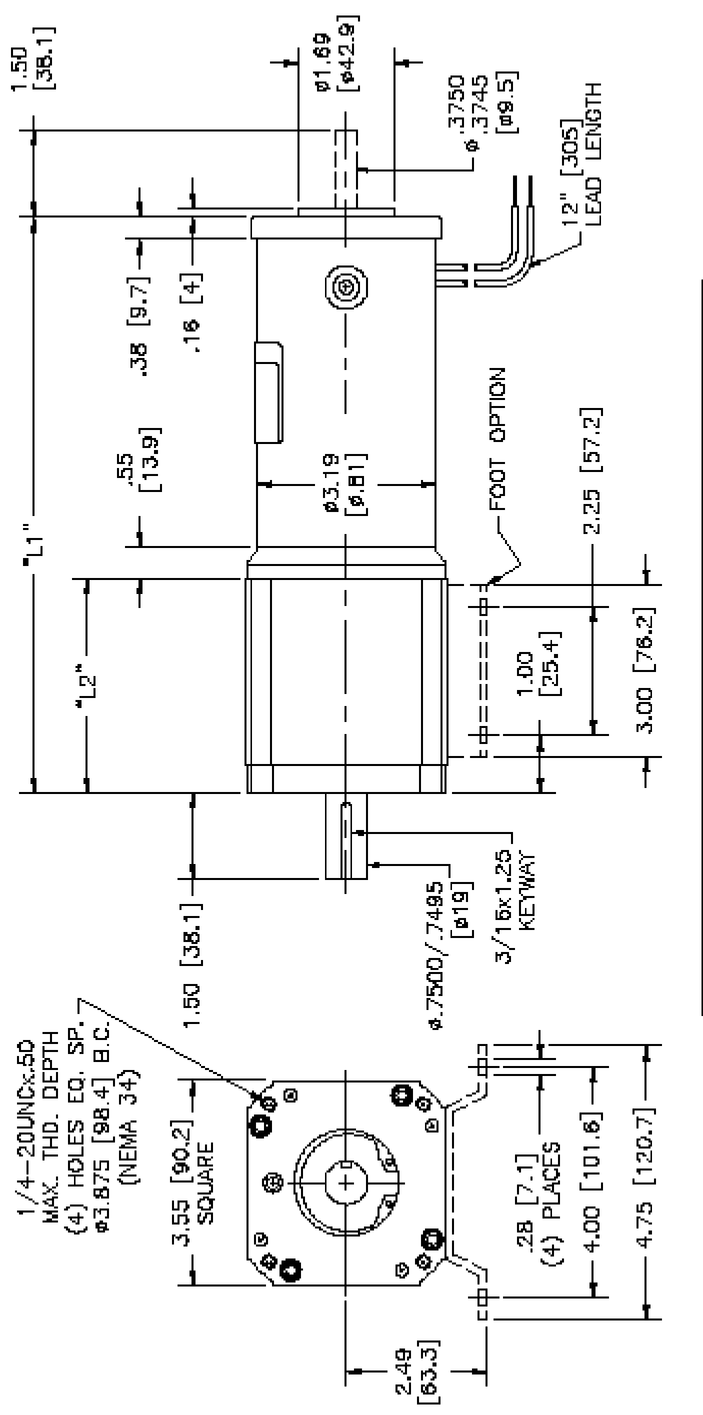
Current:	90.31 amps
Torque:	1146.86 lb-in

Constants:

Ke:	4.53 v/krpm
Kt:	0.38 lb-in/amp

GROSCHOPP™

Groschopp, Inc., 420 15th Street NE, Sioux Center, IA 51250-2100 USA
(712) 722-4135 Phone · (800) 829-4135 Toll Free · (712) 722-1445 FAX
www.groschopp.com



MODEL	RATIO	"L1"	"L2"
PM8014-PL73__	5:1 – 61:1	9.99 [254]	3.73 [95]
	90:1 – 377:1	10.60 [269]	4.34 [110]
PM8018-PL73__	5:1 – 61:1	11.49 [292]	3.73 [95]
	90:1 – 377:1	12.10 [307]	4.34 [110]

NOTES:
1. ALL DIMENSIONS ARE NOMINAL.
2. [mm] = MILLIMETERS.

GROSCHOPP
ELECTRIC MOTORS • REDUCERS • GEAR MOTORS • MOTOR SETS

PM801_ _–PL73_ _
CATALOG STANDARD
REV: 1
DATE: 12-18-01

MOTORTEC™ DATA SHEET

Customer:	Groschopp.com	Quote No:	---
Application:	---	Date:	07/09/08
Prepared by:			

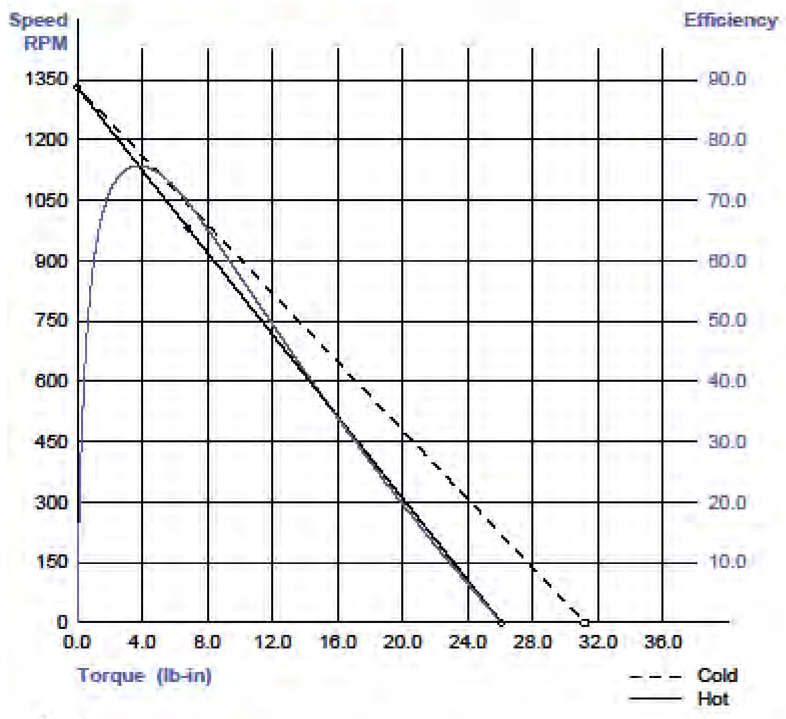
Motor:	PM 8018
Voltage:	12 v DC
Speed:	950 rpm 2 Poles
Gearbox:	None

Catalog Part No: **4106**

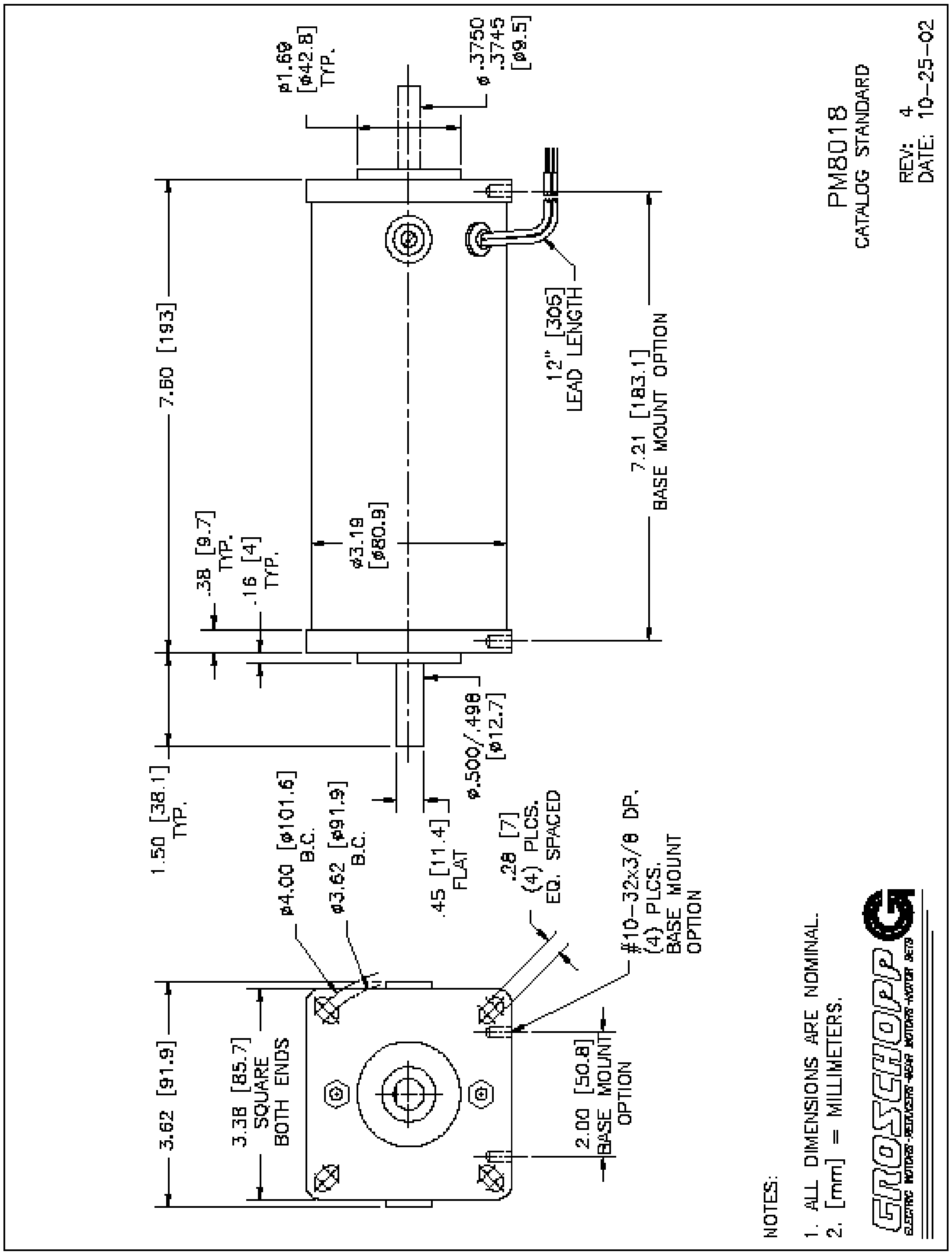
SPECIFICATIONS

Rating:		
	Speed:	982.0 rpm
	Torque:	6.8lb-in
	Current:	9.81 amps
	Output:	79 watts
	Output:	0.1060 HP
Duty Cycle:		
	On:	Continuous
	Off:	
Efficiency:		
	Gearbox:	
	Motor:	69.7 %
	System:	69.7 %
Start/Stall Conditions:		
	Current:	41.79 amps
	Torque:	31.20 lb-in
Constants:		
	Ke:	9.03 v/krpm
	Kt:	0.76 lb-in/amp

Speed, Efficiency Vs. Torque



Notes:


3.62 [91.9]
3.38 [85.7] SQUARE BOTH ENDS
2.00 [50.8] BASE MOUNT OPTION
ø4.00 [ø101.6] B.C.
ø3.62 [ø91.9] B.C.
.45 [11.4] FLAT
.28 [7] (4) PLCS. EQ. SPACED
#10-32x3/8 DP. (4) PLCS. BASE MOUNT OPTION
1.50 [38.1] TYP.
ø.500/.498 [ø12.7]
.38 [9.7] TYP.
.16 [4] TYP.
ø3.19 [ø80.9]
7.60 [193]
7.21 [183.1] BASE MOUNT OPTION
12" [305] LEAD LENGTH
ø1.69 [ø42.8] TYP.
ø.3750 .3745 [ø9.5]
NOTES:
1. ALL DIMENSIONS ARE NOMINAL.
2. [mm] = MILLIMETERS.
GROSCHOPP
ELECTRIC MOTORS · REDUCERS · GEAR MOTORS · MOTOR SETS
PM8018
CATALOG STANDARD
REV: 4
DATE: 10-25-02

MOTORTEC™ DATA SHEET

Prepared by: Groschopp.com 01/06/09

Product ID:	50102

Motor:	PM 6013	
Voltage:	12 v DC	
Speed:	900 rpm	2 Poles
Gearbox:	PS 1900	120:1 Ratio

SPECIFICATIONS

Rating:

Speed:	7.6 rpm
Torque:	113.3lb-in
Current:	2.9 amps
Output:	10 watts
Output:	0.0137 HP

Duty Cycle:

On:	Continuous
Off:	

Efficiency:

Gearbox:	61.4 %
Motor:	51.9 %
System:	31.8 %

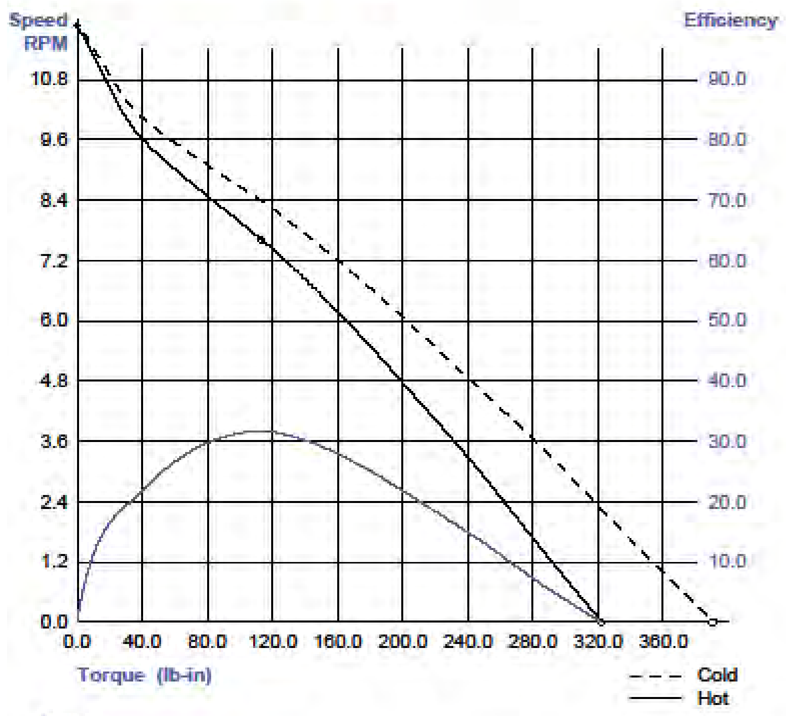
Start/Stall Conditions:

Current:	7.31 amps
Torque:	390.75 lb-in

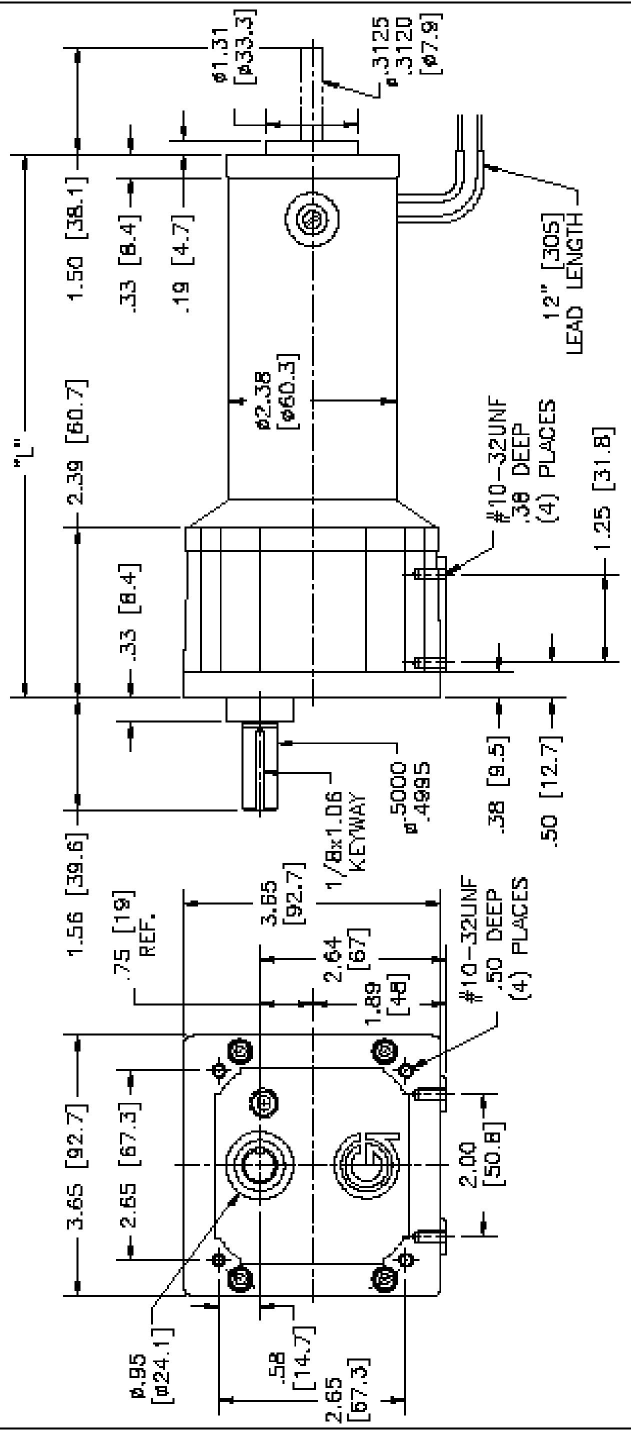
Constants:

Ke:	7.82 v/krpm
Kt:	0.66 lb-in/amp

Speed, Efficiency Vs. Torque



Notes:



MODEL	"L"
PM6013-PS19__	6.81 [173]
PM6015-PS19__	7.64 [194]

NOTES:
1. ALL DIMENSIONS ARE NOMINAL.
2. [mm] = MILLIMETERS.

GROSCHOPP
ELECTRIC MOTORS • REDUCERS • GEAR MOTORS • MOTOR SETS

PM601__-PS19__
CATALOG STANDARD

REV: 2
DATE: 11-6-06