Spring 5-2006

IEEE Robot Design

Brian Scott Burress
University of Tennessee-Knoxville

Justin Wayne King
University of Tennessee- Knoxville

Andrew Northam Wooster
University of Tennessee- Knoxville

Project CRAIG
2006 IEEE Hardware Design
Senior Honors Project

Brian S. Burress
Justin W. King
Andrew N. Wooster

April 28, 2006

Advisor: Dr. Mongi Abidi
Department of Electrical and Computer Engineering
The University of Tennessee, Knoxville

Table of Contents

Overview 3
Competition Rules 4
Chassis 8
Power System 8
Batteries 8
Voltage Regulation 10
The Drive System 13
Motor Controller 13
Motors 15
The Claw System 19
The Claw 19
The Collapse System 20
Modified Servos 21
Belt Systems 25
Package Lift System 25
Top Belt System 27
Loading/Unloading 29
Bin Expansion/Retraction: 29
Storage Bins 33
Solenoids 35
Unloading pins 36
Processing and Sensors 37
Flow Chart 43
Conclusion 49
Appendix A: Track Layout 51

Overview

Challenge: FedEx has three flights about to leave from Memphis. One will depart in 3 minutes, one in 4 minutes, and the last in 5 minutes. Twelve FedEx packages must be loaded correctly on the three planes before they depart.

Test Track: The airport ramp area is represented by a sheet of plywood. Package-sorting/loading vehicles start from its parking space (a square painted in one corner of the ramp) upon a verbal signal that begins each round of the competition.

A package stacking chute consisting of a triangular FedEx mailing tube positioned vertically over one corner of the board is filled with 12 "packages." Barcodes affixed to each of the packages indicate the plane onto which it should be loaded.

The airplanes are represented by cardboard boxes with open tops. They are placed on the ramp in a configuration similar to that shown in the ramp layout drawing.

Approach: An autonomous package loading robot will extract packages, one at a time, from a stack inside the package chute. As each package is removed from the bottom, the next package drops into position onto the ramp surface until all packages have been selected. The order of the packages coming from the chute is unknown to the robot for each round.

The robot will read the barcode affixed to each package to determine the airplane onto which it should be loaded. Each plane has four packages assigned to it. It is left to each team's design strategy how to optimally get the packages onto the correct airplanes. Examples include pre-sorting, loading each package in turn on the correct plane, etc.

Points will be awarded for the timely and accurate loading of packages and deducted for errors or damage.

Competition Rules

Ramp:

1. The board is 3/4" plywood, cut into a 4' x 4'square.
2. The board's surface is painted black, while all markings on the board are painted white.
3. There is a starting square 8" x 8" in one corner, into which the robot must fit at the start and finish of each round.
4. Rectangles are painted to represent airplane parking spots, slightly smaller (5" wide by 11-1/2") than the size of the airplanes.
5. Lines lead from the starting square to the package stacking chute and to the planes.
6. There is a "positioning mark" perpendicular to the lines leading past each plane which is aligned with the end of the plane.
7. There is a large unmarked area to the side of the package chute which may be used by a team at its discretion for additional working room.

Planes:

1. Airplanes are represented by cardboard boxes measuring 6" wide by 12" long by 3" high, and can be made by cutting a 6" x 6" x 12" box (such as OfficeMax item #172873) in half.
2. The boxes are left open on top.
3. Although a plain brown cardboard box is described, the actual boxes used may be of any color.
4. The airplanes are numbered 1, 2, and 3 and will be positioned over their corresponding rectangles painted on the ramp. The planes will not be identified in any way with their number.
5. The boxes will not be affixed to the ramp other than by gravity.
6. Airplane 1 leaves three minutes after the start of the round. Airplane 2 leaves four minutes after the start. Airplane 3 leaves five minutes after the start.
7. At the time of departure for each plane, it will be physically removed from the playing field.
8. Planes must be loaded from the "loading zone" side of the plane. This is the side of the plane closest to the starting square side of the board. (This is depicted in the drawing showing the board layout but is not painted on the actual ramp.)

Packages:

1. Packages are cut from a length of dimensioned #2 pine lumber nominally 2" x 4" (approximately 1-1/2" x 3-1/2") cut into twelve blocks each 2-1/8" long. (Save the rest of the board.) Edges may be sanded for smoothness, but will not materially change the overall dimensions of each block.
2. The blocks may be painted any color or left in the natural finish of the wood.
3. Each package will have a barcode label on the top and on the front (side facing the robot) as delivered by the chute. Both labels are oriented parallel to the long edges of the package.
4. Barcode labels will be white with printing in black.
5. The barcode format is Codabar. Registered teams will receive a sheet of actual labels for project development.
6. There are a total of twelve packages in a round, four of which are designed for each plane.

Robot:

1. The robot must be a single autonomous device.
2. It may not separate into multiple units.
3. The maximum starting and ending size of the robot is 8" wide by 8" long by 12" high.
4. Upon starting, the robot may expand to a maximum size of 14" wide by 14" long by 20" high.
5. Upon completion of the round, the robot must again be no larger than 8" wide by 8" long by 12" high

Package stacking chute:

1. The package stacking chute will consist of a triangular "FedEx Tube" measuring nominally 6" on each side and fastened vertically in one corner of the ramp.
2. Both ends of the tube are open, which requires cutting the flaps off to an even edge on all three sides. Cut the tube to half its length. (This is more easily done before assembling the tube.) (Be careful when gluing the overlapping sides of the tube together so that the finished interior dimension along that side is sufficient for the packages to fit snugly, but with enough tolerance to slide down the chute! Careless assembly can result in inconsistent overlap, causing binding or slop in the completed chute.)
3. The lower end of the chute is between 1-3/4" and 2" above the ramp surface. (To fix the tube in this position, cut the remaining length of 2" x 4" lumber in equal lengths and fasten (staples recommended) onto two sides of the tube. The fasteners should be from the inside of the tube into the wood. (Do this before you glue the tube into its triangular configuration.) Mount the wooden supports to the

ramp using four 1-1/2" x 1-1/2" corner brackets and sixteen #8 x 3/4" flat-head wood screws.

4. The chute will be loaded with twelve packages for each round. The order of the packages will be the same for every robot in a round, but it will be different in each round. (The easiest way to load packages into the chute and maintain the same orientation is to insert a stick, longer than the chute is high, against the side that the robot will approach. Insert each package between this stick and the side, pressing down just enough to move the stack of packages a bit farther down the chute. Do not let the packages drop to the bottom on their own. By maintaining pressure against the stack with the stick, you can hold the entire stack of packages in position until the last one is loaded, at which time you ease off the pressure to allow the entire stack to slowly drop to the bottom of the chute.)

Administration:

1. To qualify for the contest, a robot must extract one package from the loading chute, allowing the next package to fall from the shoot onto the ramp. A team will be given up to three rounds to qualify, with each round lasting two minutes.
2. The starting and ending size of the robot will be confirmed for each round by placing a box over the robot. Each team will perform the measurement of its own robot under the supervision of the Contest Committee. The measuring box must touch the board surface on all sides.
3. All robots competing in a round must be positioned in a holding area prior to the beginning of that round. Electric power will be available. While it may be powered, charging, or turned off while awaiting its run, a robot may not be touched by the team until its allotted start time. It will be returned to the holding area until that round is completed by all robots competing in that round.
4. No programming of the robot will be permitted once the order of the packages has been revealed for a round.
5. An audible signal (the word "GO") will be given by the Contest Committee to start
   each round. Simultaneously, the timing will begin. Upon this starting signal, each team will manually activate its robot.
6. The robot will return to the required dimensions and turn on a blue LED to signal the completion of its round.
7. Elapsed time will be recorded from the starting signal until
   a. the robot signals completion or
   b. a time limit of 6 minutes has expired.
8. If the robot runs off the board, a time of 6 minutes will be recorded.
9. If a team picks up its robot prior to the completion of its autonomous round, a time of 6 minutes will be recorded.
10. If the robot does not return to the required dimensions at the completion of its round (either by signaling or by elapsed time), a time of 6 minutes will be recorded.
11. The order of competition for each round will be randomly determined.

Scoring:

1. Each robot will compete in three rounds.
2. The maximum number of obtainable points for all three rounds is 300.
3. Points are awarded as follow:
    a. 8 points for each package placed on the correct plane by its departure time
    b. All packages have the same point value.
4. Points will be deducted as a penalty for each occurrence of the following:
    a. Bump a plane enough to move it out of position (white rectangle visible) – minus 12 points
    b. Package loaded on the wrong plane – minus 2 points
5. Packages must be "on" an airplane at the time of its departure to count. If a package once loaded falls out of the plane, it is considered left on the ramp and is not scored.
6. If the robot leaves the board, its round ends.
7. If a plane is bumped, it will be manually repositioned once the robot leaves the vicinity of the plane.
8. The total time elapsed for each of a robot's three rounds will be used in the event of a tie, with the faster robot winning.
9. If a robot damages a package so that it cannot be re-used, we will be upset. However, any robot that can mangle a block of wood is not one we would want to have mad at us, so no points will be deducted.

Chassis

In the design of the chassis we had to first determine what material we wanted to build our robot with. Many options were available and we researched them through various sources including department professors, vendors, and the Internet. Two main possibilities had our focus, aluminum sheets and expanded PVC board. Aluminum was a viable option because it could be cut and shaped into whatever form we needed it. It was strong, somewhat lightweight, durable, and it looks great. On the other hand we evaluated expanded PVC board. Expanded PVC board is strong enough for our purposes due to its high stiffness, especially the thicker pieces, it is extremely lightweight, and it is durable. It is resistant to both moisture and many chemicals. It is also easily shaped and cut. The board can be shaped much like wood. Bending of the board is also possible by boiling the board for 10-15 seconds. After boiling the material is malleable, and will harden when cooled. Probably the biggest advantage of expanded PVC board however, is the low price. We were able to buy a 12"x12" board for only $4.49. For these reasons, we decided upon the expanded PVC board. The particular expanded PVC board we purchased is called Sintra board.

Figure 1: Sintra Board

Power System

Batteries

One of the primary design considerations of any power system is the means by which power is generated. In the case of our robot design, the most obvious source of power is a battery or combination of batteries. When choosing batteries for an application such as our robot design, there are many possible battery types and configurations that can be used. One of the most important characteristics of the batteries used for our overall design was that they should be rechargeable. To gain more understanding of the most common rechargeable battery types, research was done to identify them. From this research we were able to evaluate the characteristics associated with those battery types and eliminate from design consideration those that were not practical for our application. **Table 1** lists the most commonly available rechargeable battery types along with information that was most important in determining those that were most practical for our robot design.

Rechargeable Battery Comparison Chart

Battery Type	*Features*	*Disadvantages*	*Applications*	*Nominal Cell Voltage (V)*	*Output Rating (Ah)*	*Charge Time (hours)*	*Price ($)*
Lead-Acid	most commonly used rechargeable, high power/weight ratio	low energy/weight ratio, must be kept upright	automotive, DC motors, emergency lighting, wheelchairs	2	1-35	8-16	25
Nickel-cadmium (NiCd)	used as replacement for alkaline batteries, high energy/weight ratio, keep near constant voltage throughout life, suited for high-current applications	more cells required than other battery types, hard to detect when battery power is low	portable electronics, toys, power tools, wireless phones, electric cars, start batteries, RC cars	1.25	3	1-2	50
Nickel metal hydride (NiMH)	environmentally friendly, high capacity, good for high drain applications	lower energy density, not good for fast discharge rate applications, charging must be processor controlled, operation sensitive to temperature	hybrid/electric vehicles, digital cameras	1.25	1-5	2-4	60
Lithium-Ion	do not suffer from memory effect, low self-discharge rate, ideal for low-voltage/low-current electronics	not as durable as other types, should be kept cool during operation, can be dangerous if mistreated	notebook computers, cellular phones	3.6 (4.2V charging voltage)	0.5-3	2-4	100
Lithium-Ion Polymer	can be lighter than other types, high energy density, long run times	load must be removed if battery falls below 3.0V or it will not accept charge, battery specific chargers required	mobile phones, RC aircraft, PDAs, laptops	3.6	0.5-3	2-4	100

Table 1: Rechargeable Battery Characteristics

Based upon the information found in Table 1, it was determined that the most reasonable options available were the lead-acid, nickel-cadmium (NiCd), and nickel metal hydride (NiMH). The lithium-ion and lithium-ion polymer batteries were not considered primarily because of their price and sensitivity under high power conditions such as those our robot would be experiencing. The lead-acid batteries were eliminated upon further research due to their size and weight, both of which are crucial in minimizing in our design. Nickel metal hydride batteries were ultimately chosen because of their ability to discharge large amounts of current when configured as multi-cell battery packs and their ease of use. Two such nickel metal hydride battery packs were already available, so no expense had to be incurred from obtaining these. Each battery pack is a 6-cell rectangular configuration for a nominal 7.2V output. The amp-hour rating is approximately 2-3Ah, but the battery packs are capable of producing enough continuous current to handle the maximum current draw of our robot at 6.35A while in motion. The battery packs are configured in series for a total voltage of 14.4V, which is enough to provide both the 12V rail needed by all of the DC electromechanical devices as well as the 5V rail needed for all the digital components in our design.

Voltage Regulation

After determining the battery type to be used in our robot design, a method of providing regulated 5V and 12V rails had to be determined. To accomplish this task, the simplest way is to implement linear voltage regulator integrated circuits to step down the 14.4V provided by the NiMH battery packs to 5V and 12V. However, this method poses some problems with providing a regulated 5V rail. A 5V linear voltage regulator would have to step down 9.4V from the supply voltage, creating a large amount of power dissipation across the regulator. As a result, special care must be taken to ensure that the

IC does not overheat by providing a heat sink. This configuration takes up valuable space and also creates heat that can affect other components of our robot system.

As an alternative to using a linear voltage regulator IC to provide the 5V rail, a dc-dc voltage converter can be implemented. For our design, this was determined to be the most practical option, and an appropriate component was chosen that is used very frequently in similar applications. **Figure 2** is an image of the dc-dc converter chosen for our design. This converter is the PW-200-M from mini-box.com. This converter requires a 12V input and can provide up to 200W of output power (dependent on the input supply) with over 95% efficiency. It requires no and takes up a small amount of space, which were ideal characteristics considering our design. ***Table 2*** provides a detailed list of the available voltage outputs from this device.

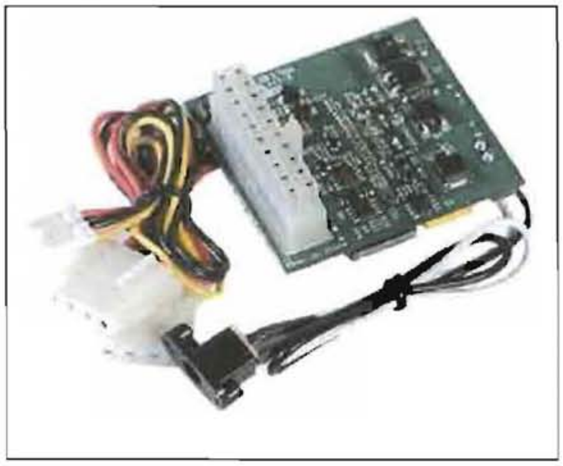


Figure 2: PW-200-M dc-dc Converter (image courtesy of mini-box.com)

Volts (V)	Max Load (A)	Peak Load (A)	Regulation %
5V	6A	10A	+/- 1.5%
5VSB	2A	10A	+/- 1.5%
3.3V	6A	10A	+/- 1.5%
-12V	0.1A	0.2A	+/- 5%
12V	12A	13.5A	Switched input

Table 2: dc-dc Converter Voltage Outputs (information courtesy of mini-box.com)

Based upon the information about the dc-dc converter listed in Table 2, it was determined that it met all the necessary power requirements for the output voltage levels of our robot. The 5V and 12V outputs were of most interest, and the available load currents that can be drawn from the device are well within the range required of the components for our robot. The highest single component current draw for the 5V rail is the PC/104 mainboard that draws around 2.5A of current, and the highest system current draw for the 12V rail is the drive system containing the stepper motors that draws approximately 3.75A while in motion. Table 3 provides an inventory of power requirements for every electrical component in the design.

System		Required Current	Required Voltage	Priority*
	Components			
Drive System				
	Controller	0.6A	12V	1
	Stepper Motors	3.2A	12V	2
Controls				
	PC104 Board	2.5A	5.0V	1
	6812 I/O Board	0.05A	5.0V	1
Package Extraction				
	Claw Servo	0.5A	5.0V	3
	Claw Motor	1.0A	12.0V	3
	Small Arm Servo	0.3A	5.0V	3
Expanding Bins				
	Side Servo	0.3A	5.0V	4
	Rear Servo	0.3A	5.0V	4
	3 Mini Servos	0.9A	5.0V	5
	Solenoids	1.0A	12.0V	6
Conveyor Belts				
	Vertical Servo	0.3A	5.0V	7
	Top Servo	0.3A	5.0V	7
Scanner				
	PosX XI1000	0.075A	5.0V	7

Table 3: Robot Design Power Inventory
***Priority 1 system is operational at all times, all other priority systems run at separate time intervals**

Because the battery packs used in our system provide a nominal 14.4V output voltage and the PW-200-M dc-dc converter requires a 12V input, it was necessary to provide voltage regulation down to 12V from the 14.4V supply. There are a number of possible options that were possible for accomplishing this, but the most practical for this application is the use of a 12V linear voltage regulator. Because only about 2.4V must be stepped down, very little heat and power dissipation would be required of an integrated circuit to step down the voltage. Because of the high current draw necessary from our robot at its peak current draw condition, it was necessary to find an IC that was capable of handling this current. A part made by Linear Technology called the LT1083-12CP was found to be able to handle the power requirements needed. This device is capable of handling a maximum output current of 7.5A at +12V. This is capable of providing the 6.35A of current necessary by our robot during its peak current draw period. To implement this solution, a circuit was designed to take the input voltage from our batteries and output the necessary +12V that would then be fed to the PW-200-M. **Figure 3** shows a detailed schematic of this circuit. The circuit is not very complex, but special care had to be taken to ensure that the capacitor selection provided the necessary noise and ripple suppression required both at the input and output of the device.

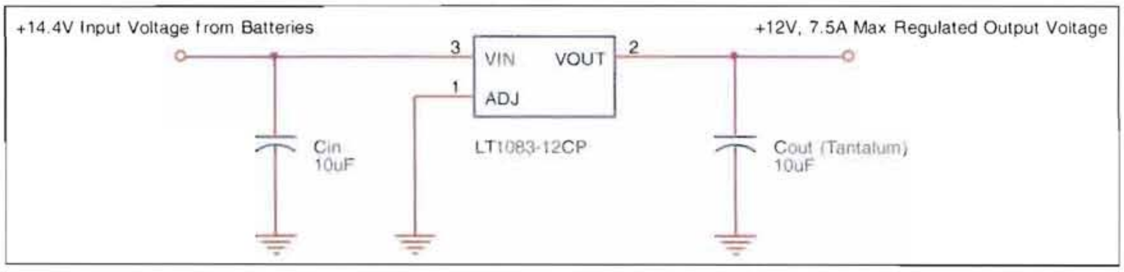

+14.4V Input Voltage from Batteries
+12V, 7.5A Max Regulated Output Voltage
3
VIN
VOUT
2
1
ADJ
LT1083-12CP
Cin
10uF
Cout (Tantalum)
10uF

Figure 3: Schematic of +12V Regulation Circuit

The Drive System

Motor Controller

When designing the drive system, there were two main options to be considered. One option involved using DC motors with gear housings to provide more torque, and the other option involved using stepper motors. Both options were found to be feasible; however, based upon the requirements of the competition it was decided that additional precision that could be provided by a stepper motor drive system was desired. As a result, research and testing was performed on stepper motors as well as controllers available for them. As a start, it was necessary to find a controller that would allow the simultaneous controlling and operation of two stepper motors. There was difficulty finding available controllers, but one made by Peter Norberg Consulting, Inc. was found to be perfect for our application. **Figure 4** is an image of the stepper motor controller board used in our robot design. It is the BiStep2A from stepperboard.com, and it can provide a maximum current of 2Amps/phase per motor. When operated in "double current mode", it can provide up to 4Amps/phase to a single stepper motor. This controller can accept 2 different voltage supplies, one for the stepper motors and one for the control logic.

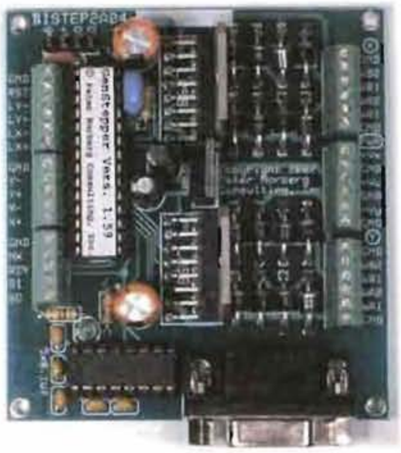


Figure 4: BiStep2A Stepper Motor Controller

The maximum allowable motor supply voltage is 45 volts, while the maximum allowable logic supply voltage is 15V. For efficient design it was determined that the best option would be to provide the stepper motor supply voltage and logic voltage from the same supply, and it would be drawn from the +12V power rail.

The stepper controller provides full control of the 2 stepper motors used in our drive system by accepting commands through an RS232 serial interface related to speed of stepping, slew direction, motion based on absolute position, etc. When commands are administered by the PC104 board through this interface, control of the motors becomes very streamlined. One of the most important commands that is issued for use in our application involves modifying the run rate target speed for a selected motor. When coupled with the IR sensors beneath the robot, using this particular command is what makes the line-following possible in our design. The BiStep2A controls stepper motors through microstepping, and 16 microsteps is equivalent to one full step of the motor. The motors used in our design require 200 steps for a complete revolution, so the default value of 800 microsteps/second (50 full steps) provides rotation of the motor shaft at 0.25 revolutions/sec. By simply increasing this default value of 800, it is clear how the speed of our robot can be modified to meet the time specifications of the competition.

Motors

After deciding on the use of stepper motors in our robot drive system, it was necessary to determine the torque required of the system during competition. To gain understanding on the operation of the stepper motors and how their torque characteristics affect operation, 2 stepper motors that were readily available were tested as a benchmark. The motors tested were 8.4V with 30ohms/winding used in a bipolar configuration. These motors drew approximately 300mA of current while in motion; however, their maximum holding torque was only rated at 11 oz.-in. This was proven to be very insufficient for our design, as the motors would slip when a load of approximately 1.25lbs. was placed upon the prototype drive system as seen in **Figure 5**.

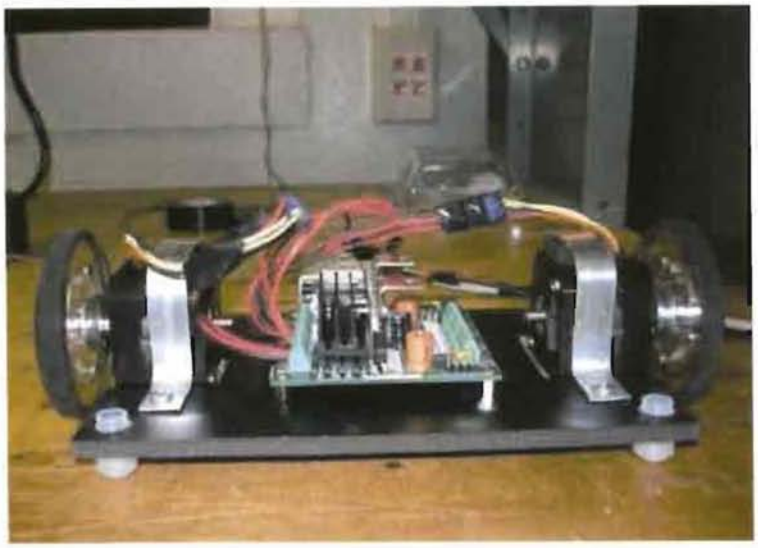
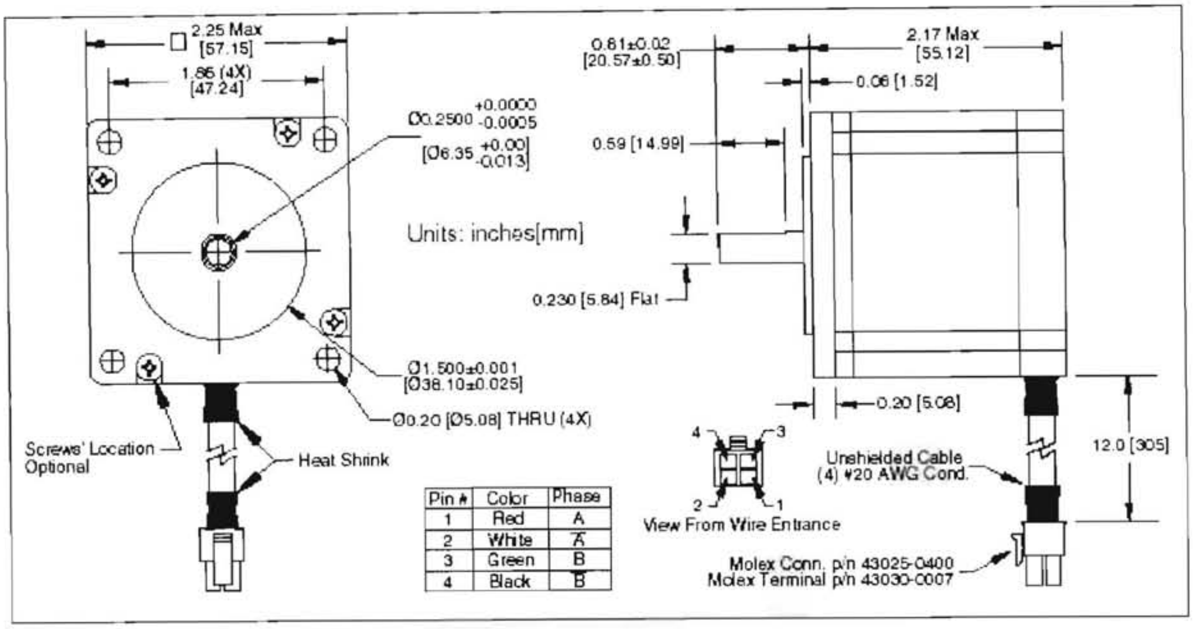


Figure 5: Prototype Drive System

It was calculated that the stepper motors used in our design would have to provide enough torque to move up to 10lbs of components and packages at full load. Two such motors, the STP-MTR-23055 motors from automationdirect.com, were available that have a holding torque rating of 166 oz.-in of torque, more than enough to handle the requirements of our design. There were a couple of problems with the use of these motors, however, that had to be considered when integrating them into our design. The motors are rated at 2.8A/phase – a value that our stepper controller is not capable of supplying without failing. The motors are also NEMA size 23, compared to a preferred size of NEMA size 17. **Figure 6** provides the dimensions for the motors used in our drive system.



Pin #	Color	Phase
1	Red	A
2	White	$\overline{A}$
3	Green	B
4	Black	$\overline{B}$

Figure 6: STP-MTR-23055 Motor Dimensions (image courtesy of automationdirect.com)

These stepper motors have a winding resistance of approximately 1.5ohms. When used in combination with our 12V supply rail, the motors would force pulling their rated current. To eliminate this behavior, research was done to determine the practical use of a series winding resistance in addition to the inherent winding resistance of the motors. By adding power resistors in series with the motor windings (power resistors were necessary because the power dissipation is considerably high approaching 10W), a current draw of approximately 1.6A/per phase was achieved. This allowed us to provide a high amount of current to each motor to obtain necessary torque while also not pushing the limits of our stepper motor controller. An advantage to providing a series resistance with the motor windings is that the electrical time constant of the system is reduced. The following formulas show how this is possible:

$$T_{elec} = L/R_{total}$$

$$R_{total} = R_{series} + R_{phase}$$

where T_{elec} = electrical time constant

L = phase inductance

R_{series} = current limiting series power resistance

R_{phase} = winding resistance of motor

By reducing the electrical time constant, the current in the winding rises faster, and a faster response from the motors can be seen when issuing commands from our stepper motor controller. This reduction of time ultimately increases the accuracy of our line-following system because there is less of a time delay response from sensor detection to motor speed change. The only downside to this configuration is the high power dissipation in the series power resistors; however, because the duration of the competition does not exceed 6 min. for a given round, this dissipation is not considered to be detrimental to the battery life during that time.

The other disadvantage to using these motors was their large size. Some modifications to other components in our system had to be made, most particularly in the extraction claw design, but in the end we were able to design for the larger size of the STP-MTR-23055 stepper motors.

The Claw System

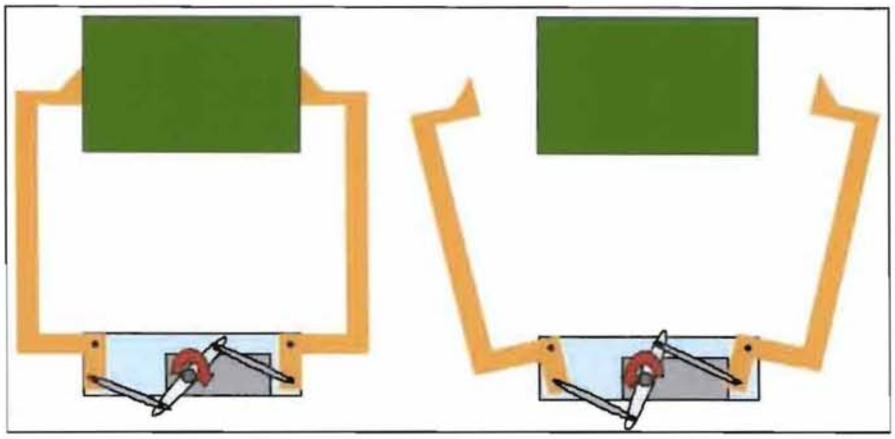
To grab the packages from the chute we decided to use a claw. Another solution would have been to use a system with wheels which, by rotation, pushed the packages out of the chute. But we realized that it may have been less precise than a claw.

Our system had to be able to grab, hold and pull back the package. The structure also had to allow the claw to spread and contract entirely in the robot to fit the size constraints. We divided it into two different functions. The first one is the claw itself, and the second one is the collapsing system.

The Claw

There are a lot of claw designs used in a wide variety of purposes. In our case the claw had to be enough powerful to hold the first package with the eleven others stacked on top of it. However, the claw body had to be as small as possible to leave space for it to move back and forth. We decided to use a servo motor to drive the claw. Indeed, the servo motors have an excellent precision and a wide range of torque. We used a Futaba servo. It has a 5kg/cm torque (70 oz/in) which fit our specifications. The claw design can be observed in **Figure 7**.

Figure 7: Claw Design



We positioned the servo motor in order to create more space. The form of the claw was made to allow it to go around the lift system. The material we used was plastic; however, because the design was pretty thin, a metallic structure was added to increase its rigidity. In order to improve the grip soft rubber was fixed on the part that would contact the package.

The Collapse System

As with the claw, there is a wide range of systems which allow linear movement. The one we chose is a screw system. It fits two essential specifications of our design which are a good rigidity and excellent precision. **Figure 8** shows how it works.

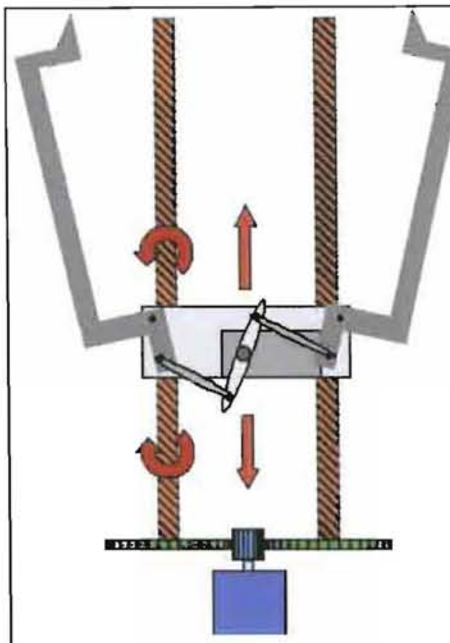


Figure 8: Collapse System for claw

Two nuts were included on the claw body. They were fixed, so when the screws turn, the claw does not turn but moves back and forth along the length of the screw. The driving system was made with a 9V DC motor and a set of gears. These gears have two purposes. First, they permit the motor to drive both screws simultaneously in the same direction. Second, they allow us to modify the speed and torque simply by changing the size of the gears. This system has been really useful in finding a good compromise between speed and torque (halving the speed gives twice the torque; not counting the loss of power due to friction). The custom gear set contains 6 gears and divides the speed by 12. The DC motor we use is a 12v motor. It is completely efficient and turns at 19,000 rpm (no load) at this voltage.

The objective was to load the packages as quickly as possible. We fixed elapsed time for a claw to make a round trip with a package at 4sec, which means approximately 50sec to load all of them.

We have a screw step of 1.25mm. The movement length is 70mm ($2^{1/2}$in) which is the minimal distance to pull a package out of the chute. A full movement of 70mm corresponds to 56 rotations (*70 / 1.25*). We can presume the motor speed will be around 15,000rpm burdened with the load. Because the speed is divided by 12, the screws rotation speed becomes 1250rpm.

So the calculation of this traveling time (T) is the following::

*T = (rotations*60)/1250 = 2.7sec.*

A round trip will last 5.4sec.

This theoretical time is not far from the measure we made on the real system. In reality it takes approximately 2.5sec for a movement, 5sec for a round trip and 60sec for the 12 packages. However, we did not account for the time for the claw to grab and drop the packages. We can assume it will take no more than 12sec for the group (1 second per block). We arrived at a loading time of 72sec which leaves us 108 seconds (1min 48sec) to get to the first plane.

<u>**Modified Servos**</u>:

During the design and construction of the robot, it became clear that there were going to be many moving parts that would require continuous rotation drive: the bins, the belts, the claw, and the drive system. Besides the drive system, which was decided to use a stepper motor, a type of motor had to be chosen for each component. The first option was a regular DC motor. A DC motor operates simply: a voltage is applied across its leads, and it moves either clockwise or counter-clockwise, depending on which way the voltage was applied to the leads. The initial concern with using DC motors was that they would not have the required torque for whichever component used the motor. However, after the research of various types of DC motors, the majority of them had a gear box available. A gear box for a DC motor is a gear system that converts the mechanical power from the DC motor into some other form of output. This means that it can either change the direction of transmission or increase torque at the sacrifice of speed. The latter is what made the DC motors an attractive option.

However, another concern arose in the use of the DC motors. Even though a motor with adequate torque could be acquired, it was another matter all together to control a motor via some form of microcontroller or on-board computer. The robot's autonomous operation was to be controlled by a chosen computer inside the robot. In the case of this project, the on-board control system used a PC/104 form-factor motherboard for the main task-handling, with a Motorola 6812 microcontroller as the analog-to-digital converter and data acquisition (I/O). As such, there needed to be a way to control the analog DC motors. A standard microcontroller could not handle the voltages and currents to be able to single-handedly control a DC motor using its input and output pins. Therefore, H-bridges have to be used for each DC motor. An H-bridge allows a microcontroller to send a control signal to the H-bridge, which then, based on the control signal, will activate two of four switches. The four switches are initially open and connected to the operating voltage of the DC motor. Based on the control signal, two of the switches will close and allow the current to flow to the DC motor. The H-bridge allows for directional control of the DC motor and even speed control if the control signal is a pulse. The required use of an H-bridge for each DC motor was the main concern with using the DC motors. Not only do they not allow for precise operation, but they also require a separate power source and take up space. In a project in which space is of primary concern, other options besides the DC motor had to be decided upon for the separate components.

The servo motor was the second option for the operation of certain components. A standard servo motor is basically a geared DC motor, but with the addition of feedback control. As such, there are three leads that require an outside connection to operate a servo: voltage in, ground, and control signal. Essentially, a servo motor allows for high precision in position and speed. It works by gearing a DC motor up to the main gear output shaft. The main gear has a potentiometer mounted underneath it that rotates along with the main gear. The logic inside the servo reads the resistance from the potentiometer and compares it to the input signal. Based on this, it will rotate until the comparison yields the proper difference. Due to this system, there is one drawback to the standard servo: it has limited rotation, usually 90° to 180°. However, the servo is usually easily modified for continuous rotation. The drawback to a modified servo for continuous rotation is that position control is lost. Most importantly, however, is that speed control is not. After analyzing the options between the DC motor and modified servo, the decision of a modified servo for the operation of various components was used since it offered more functionality for less space. Eight servo motors can be operated with 5V to each servo by a small square-inch servo controller board vs. up 12V to each DC motor plus one H-bridge per motor. However, DC motors still had their functions in high-speed operations, such as the retraction/expansion of the front claw, which had to have a high-torque and high-speed motor. With those specifications, a DC motor is a better candidate.

The modification of each servo involved two main steps. The first step is to open the servo motor and modify the main gear. The main gear contains a small, plastic stopper that prevents it from rotating a full 360°. The mechanical modification is to remove the stopper with a sharp knife or small sanding tool. This allows the gear to rotate without obstruction. The second step involved can be done a variety of ways. Essentially, the purpose of the second step is to modify the potentiometer to remain in the same position, regardless of where the main gear is turning. This "tricks" the logic chips within the servo motor into thinking that the gear is in the incorrect position and will constantly try to correct itself, producing continuous rotation. By changing the control signal, which would usually cause a change in angle of the servo, it will instead cause a change in rotation speed and direction. The modification can either be mechanical or electrical. The potentiometer can be glued down to prevent moving and modifying the main gear to not move the potentiometer, or the potentiometer can be completely removed and replaced with two resisters. Both methods will perform the same function in the modification process.

Belt Systems

Package Lift System

Once the package is extracted from the loading chute, it is placed into the front lift system, seen in **Figure 9**. This system consists of two vertical conveyor belt systems. Each is constructed using matching pulleys and rubber belts from Small Parts©. These are mounted on the front of the robot by using a 6" by 11" piece of expanded PVC.

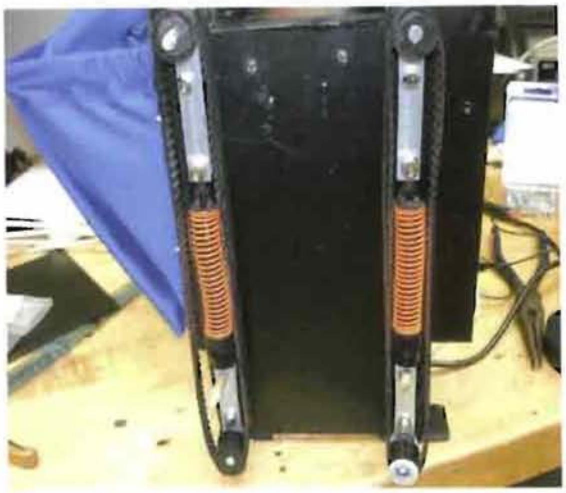


Figure 9: Lift System

Each individual belt system contains four pulleys, two on each end of the belt loop. By using four pulleys, each belt system can contain two belts, giving it a total approximate width of 1 ½". This width is needed to ensure that the contact surface area with the package is adequate to lift it to the top of the robot.

The rubber belts are each a total length of 23". When stretched around a pulley at either end, this gives a pulley to pulley length of 10 ½", which is exactly what is needed for the front lift system. The two belts on each side are connected together with cleats made from expanded PVC. These cleats are attached to the rubber belts with Super Glue©, and each is notched out so as to not rub on the pulley flanges as they go around. When the belt is running, these cleats come under the package and begin to lift it. The belts continue to run, and the package is lifted to the top of the belt system.

The pulleys are mounted on a ¼" metal rod. Originally, the pulleys contained set screws mounted on their outer edge. In order to save space on the front of the robot, the set screws were removed from the outer edge of the pulley and a hole was drilled through the middle of the pulley. The set screw was then installed within this hole. By removing the outer portion of plastic that first contained the set screw, we save approximately ¼" per pulley, or ½" total on the front of the robot, since there are two pulleys side by side.

Each belt system also consists of a shock in the middle of the pulleys. This provides a way to maintain appropriate tension on the rubber belts. Without these shocks, the belts would sag and could catch on a block as it is extracted from the loading chute and placed into the lift system. However, the longest shocks we could find were 6" when fully extended, and the front belts needed to be a total of 10 ½" fully extended. Therefore, flat pieces of metal were used to extend the shock. These pieces connect to the ends of the shock, and then extend outward to the appropriate length for the belt system. A hole is drilled in the metal pieces for the pulley shaft, and washers are used to keep the metal shaft from rubbing on the plastic pulleys.

The vertical lift system is powered by using a 7.2V geared dc motor. When powered at 4.5 volts, this provides adequate power and speed for lifting blocks. The same motor powers the two belts in order to synchronize their motions. This is accomplished by using plastic gears. By attaching a gear to each pulley shaft and then matching gears in between both shafts, the two belts are made to turn in opposite directions at the same rate. This synchronous motion is imperative to have the cleats on each belt match up when lifting a package. Otherwise, one cleat might reach the package before the other, and cause the package to be lifted unevenly or even dropped.

Finally, when the lift system has lifted the package to the top of the robot, an arm swings up and pushes the package from the lift system onto the top conveyor belt. This arm is made by using a servo motor. The motor turns 180 degrees, allowing for the appropriate amount of motion needed to transfer the block from the lift system onto the top belt. By using some of the expanded PVC, an arm was fashioned and attached to the servo.

Top Belt System

In order to move the packages into their respective bins, another conveyor belt system was implemented. This horizontal belt, **Figure 10**, that runs along the top of the robot carries the packages from the front of the robot to the back.

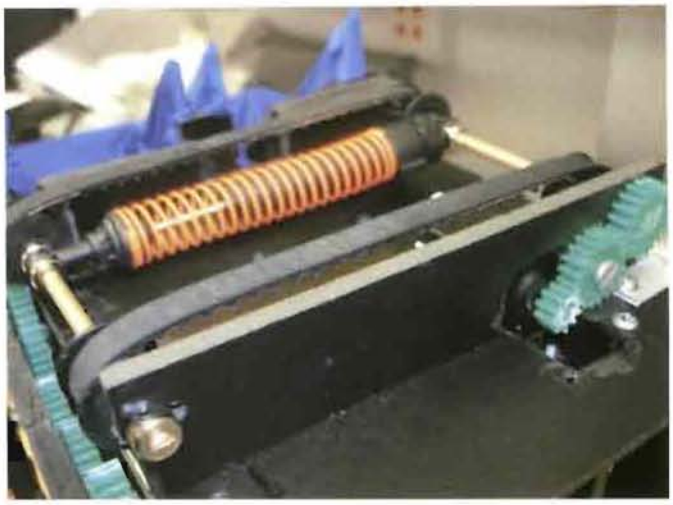


Figure 10: Top Belt System

Using 13" belts and matching pulleys from Small Parts ©, the top belt is a total of 6" long and 3 ½" wide. The pulleys used are the same as those for the vertical lift system except that the set screws have not been moved to the center of the pulley. Since this belt system did not need the space savings of moving the set screws, we left them on the outer edges of these pulleys.

For powering this belt system, a modified servo motor was used. This provided an easy interface for control as we are using a servo controller card, and also provided adequate power and speed for the belt system. The rotational speed of the shaft needed to be fast enough to load the packages in a short amount of time, but also slow enough to provide the robot time to sort the packages as they moved. A modified servo motor adequately filled all of these specifications.

The pulleys are mounted on 3 ½" long pieces of brass rod. This type of rod provided an adequate interfacing capability to the modified servo. By gluing some circle collars onto servo rotors, we were able to effectively connect the modified servo motor to the rod and turn the belt.

Two rubber belts are used in parallel for this belt system. They are spaced 3 ½" apart in order to accommodate the width of the packages.

Loading/Unloading

***Bin Expansion/Retraction*:**

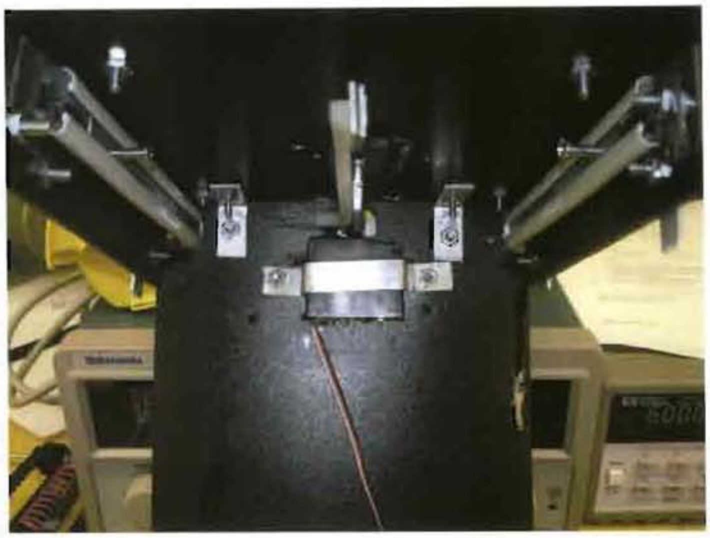


Figure 11: Bin Track System

A key part of the bin design is the expansion and retraction of each individual bin. In order to accomplish this, the top of each bin had to be made in such a way to allow a mechanism to push the bins out and pull the bins back into the robot. The bottom of each bin is a separate mechanism that is independent of the top. Initially, there were two choices: have a motorized mechanism that transfers rotary motion into linear motion or use some sort of pneumatic cylinder or linear solenoid that would do the motion in one quick move. The second idea was quickly rejected since a linear solenoid with the length of a block, which is the required amount that the bin had to expand on the top, was too difficult to find and had large power requirements. A pneumatic cylinder would have worked, but would require the use of compressed air to operate, which would mean the air supply had to be replenished at a moments notice and a solenoid valve would have to be used with each cylinder, yielding a larger, more complex system then the motorized solution. Therefore, a motorized solution was created.

The main problem with the motorized solution was coming up with a design that would allow the motor to transfer its rotary motion and torque into a linear motion with the bins. One method was to attach a rubber wheel on top of the motor, and mount it next to the top of each bin. The top of each bin would be a solid frame made out of a thin aluminum. The rubber wheel would be mounted tightly against the frame ends and allow it to retract/expand the bins. However, a problem was quickly found in that the rubber wheel would not provide enough traction to efficiently move the bins back and forth, so an alternate solution had to be found. The alternate solution involved using a gear rack and pinion combination. A gear rack is a flat piece of material that has gear teeth throughout its length. The pinion is a regular, round gear to match the gear rack. This type of system is commonly found on the steering system of cars. As someone turns the steering wheel of a car, that motion has to be transferred linearly to turn the wheels. A gear rack and pinion system is the exact solution needed to transfer rotary motion to linear motion. The pinion gear is mounted on top of the motor, and the motor is mounted right next to the gear rack. The gear rack would then be attached to the aluminum frame of the top bins. The motor would be mounted to the inside of the robot to prevent it from moving, and the pinion gear on the motor would then move the gear rack, which would cause the bins to be pushed or pulled away or back into the robot. A modified servo would be used for the bins as the motor. The modified servo provided for the easiest way to mount the pinion gear, plus offered easy directional control for retracting and expanding the bins.

The actual design used $^{1}/_{16}$'' thick by ½'' tall aluminum for the top bin frame. The aluminum was bent into a "U" shape; the open end would be on the inside of the robot, whereas the closed end would be on the outside. The bins where mounted to the robot by cutting holes into the PVC material, one for each "leg" of the "U" shape. This allowed the bins a way to move in and out of the robot body freely. A gear rack would then be attached to either of the two legs, which would provide the linear motion of the expansion/retraction as described above. A stopper would have to be mounted on the inside portion of the bin to prevent them from falling out completely. Originally, this design had one gear rack and pinion system per bin. However, because the top of the bins did not need to have separate movement, since the blocks were not going to be removed from the top, the two bins on the side where glued together and used the same gear rack and pinion system to retract/expand. Figure 1 shows the final design of the side bins.

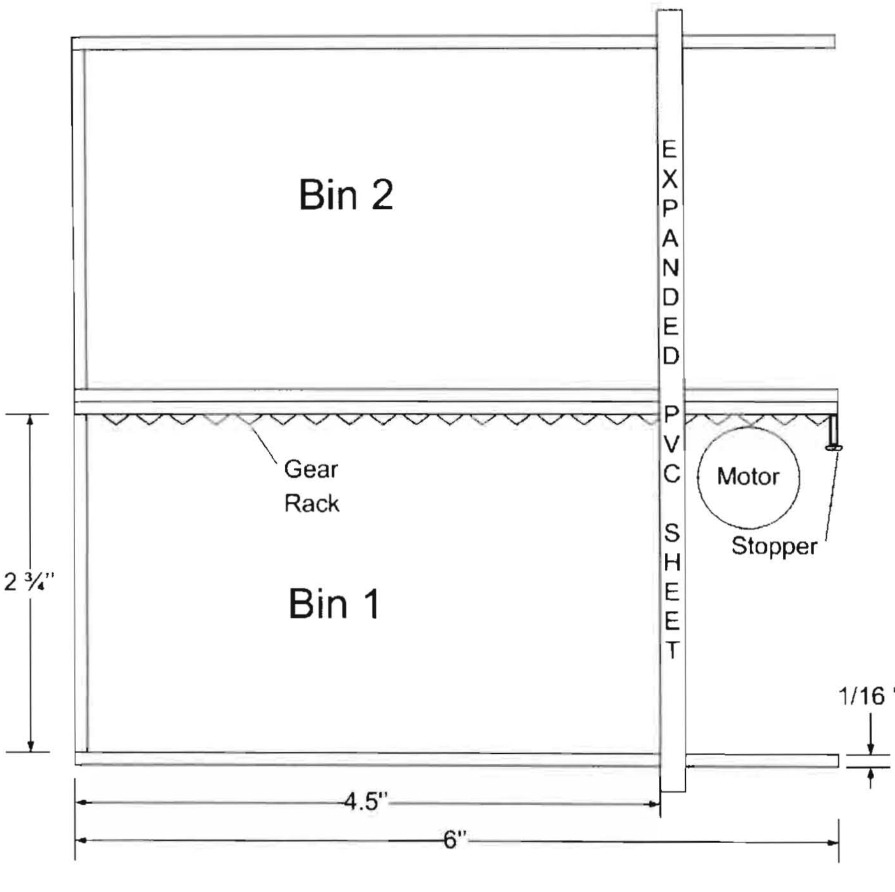


Figure 12 - Side Bins

The back bins had a similar design, except for dimensions and gear rack location. Instead of the gear rack being mounted on the inside, it was mounted on the outside to avoid the block hitting the gear rack. Figure 2 shows the back bin design.

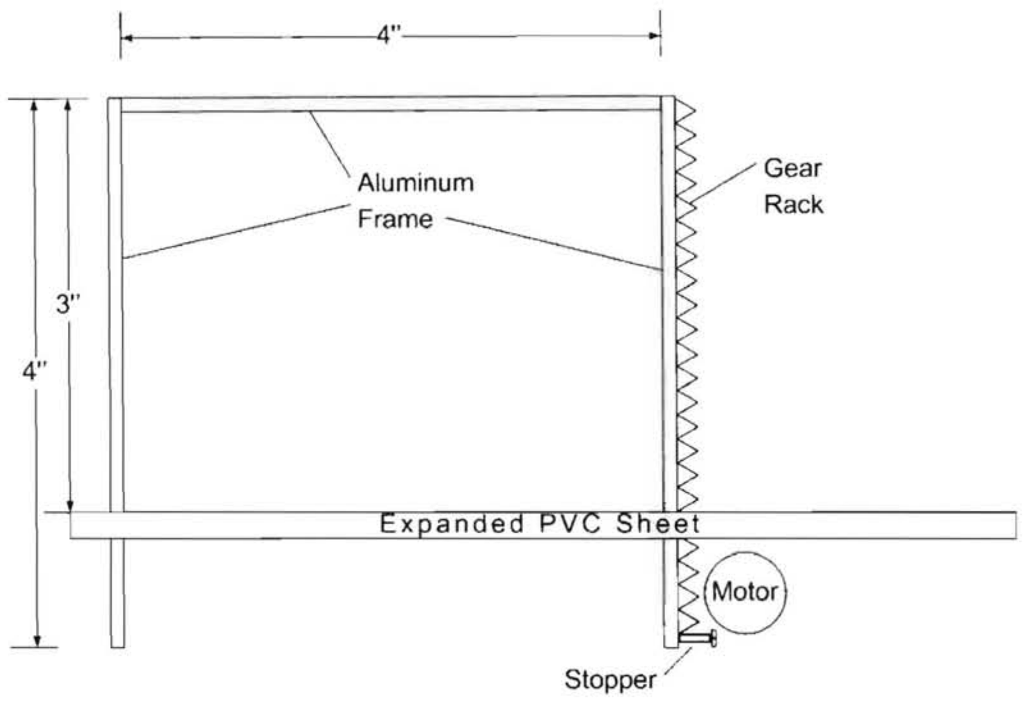


Figure 13 - Back Bin

After the initial design was completed in prototype form, there was a major problem with the movement. The frame was having too much movement inside of the robot, and it would often "derail" from the modified servo motor or expand in an odd angle. The solution to the problem involved the use of some form of guides for the legs of the top bin frames. Originally, the idea was to use small drawer slides on the legs. Since drawer slides usually use ball bearings, these would provide the benefits of a smooth and steady expansion/retraction. However, drawer slides that would meet the size constraints were difficult to find. Instead of drawer slides, the guides took the form of a modified curtain guide. These are ½'' rectangular slides for hanging curtains. Their size fit the aluminum legs of the bin frame, and, as such, they served well for guiding the bins in and out of the robot. After installing them, the bin frame would smoothly retract/expand.

Storage Bins

Since time is a crucial part of the competition, we wanted to develop a way to carry all 12 of the packages at once. The system we developed involved 3 storage bins, two on the side of the robot, and one on the back. These 3 bins corresponded to each of the 3 planes.

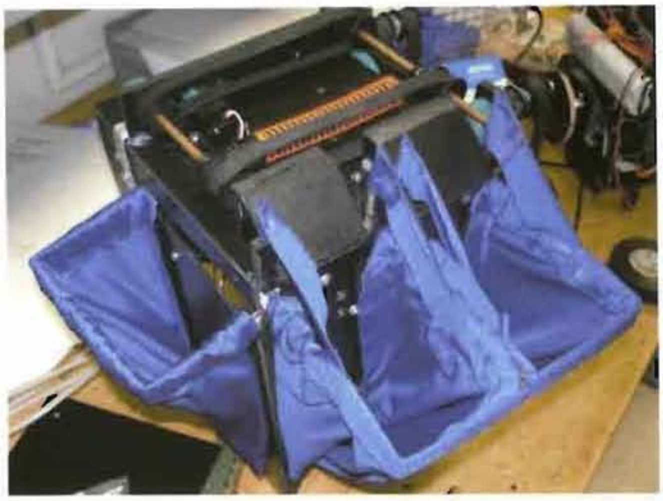


Figure 14: Storage Bins with Spandex

Due to the size constraints of the robot we determined that we needed to expand/retract the storage bins, next we had to decide what material the bins should be made of. The first choice was a solid material similar to the aluminum used to make the bin frames. After discussing this option, we were too worried about how the blocks would land when pushed into the bins. We were afraid that the block would land in a vertical position when loaded, taking up 4" of vertical space instead of 1.5". If several blocks did this, the bins would run out of room and all 4 of the blocks would not be able to fit. Therefore we decided to look for other options for the bin material.

Our next choice for the bin material was spandex. Since spandex is a stretchy material, it allowed us not to worry about how the packages land in the bins when loaded.

As the packages were loaded, the material stretched and made room for the other blocks to be loaded. The spandex was attached at the bottom of the robot, as a result when the bin was expanded the bin was slanted (**Figure 14**). This helped control how the first block landed in the bin. As the last three blocks were loaded, the added weight forced the first block to stretch the spandex and the blocks were able to lay flat on top of each other as they were loaded.

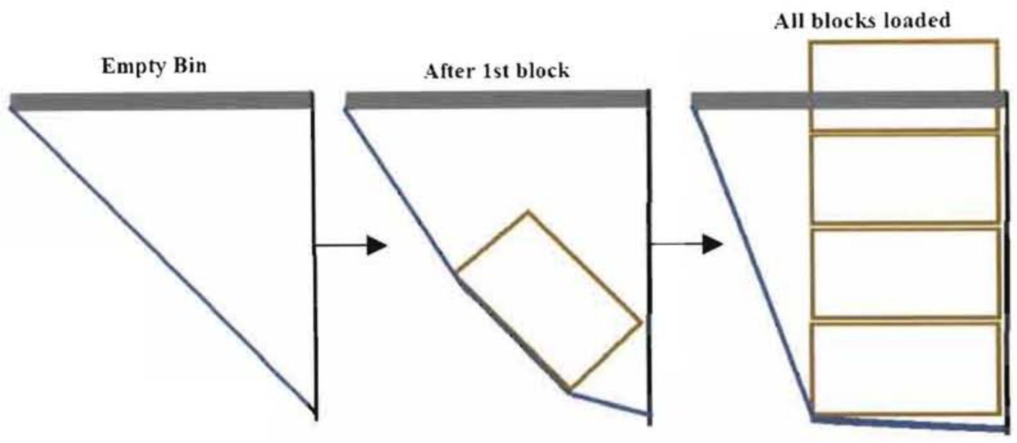


Figure 15. Loading Sequence

The spandex was connected to the aluminum bin frame by folding it around the bar of the frame and gluing it to itself. This was done so that when the bin was contracted the spandex would slide down the frame and crumple onto itself at the front of the bin. The edges of the spandex were connected to the robot using Velcro. This created the closed bins with only the top and bottom openings. The Velcro also allowed us to easily remove the spandex from the robot if the bins needed to be removed completely to make adjustments. Small pieces of spandex were connected to pins on top of the robot and then to the top of the spandex on the bins. These acted as guides and prevented the blocks from landing incorrectly when being loaded into the bins.

Solenoids

Once the package has been loaded onto the horizontal belts running along the top of the robot, there needed to be a way to force the package off of the belt and into the storage bins on the side of the robot. Due to the current height of the robot (11"-11.5") and the height constraint of the competition (12"), we had limited options on how we were going to move the bins from the top conveyor belt into the bins. We decided our best option would be solenoids. The solenoids were mounted onto the top of the robot adjacent to the belts. A push solenoid only takes a voltage that causes a pin to punch out and is retracted when the voltage is removed. The higher the voltage supplied to the solenoid the stronger the push. The length of the arm of the solenoid varies between solenoids. Our testing was done with solenoids with pins approximately .75" long. However, solenoids with pins closer to 2" long in order to ensure the block is completely pushed off the belt system (4" wide) were needed.

Only 2 solenoids are needed for our loading system, because the third bin is located directly behind the belt system. Therefore blocks that are going to plane 3 run off the back of the conveyor belt and land in the bin; they do not require a solenoid to push them off the belt. For the other two bins, sensors mounted next to the solenoids trigger when the block passes the solenoid. When the sensor is activated, if the block is supposed to go in that bin, a voltage is sent to that solenoid and the block is pushed into the bin, otherwise the block continues across the conveyor belt.

Ramps were mounted between the horizontal belt and the top of the bins to provide a guide for the block to slide down. This was done to accommodate for the distance between the top of the bin and the top belt system, which was mounted in the center of the robot in order to be centered on the claw and the front belt system. This space between the bins and the top belt was helpful because it provided a little more loading space for the blocks. If the blocks do not land in the bins in an ideal configuration, this extra space helps prevent the last block of a full bin from getting in the way of other blocks moving down the conveyor belt.

Unloading pins

We had to develop a way to release the bottom of the bins in order to unload the blocks into the plane. This release system needed: 1) to be strong enough to hold the spandex closed as the blocks were loaded, 2) to be an easy and fast release when unloading, and 3) to be small and easily mounted inside the robot. The release system (**Figure 16**) we developed involves using three servos, one for each bin. For each bin, part of a paper clip was threaded through several holes in the bottom of the spandex and then pushed into a slot cut into the robot. This paper clip acted as a "hook" for our release system. On the inside of the robot, a small "key" was cut out of aluminum and inserted into the loop of the paper clip to hold it in place. A hole was drilled into this "key" and wire was tied between this hole and a servo mounted near the top of the robot. Three Futaba servos were used for this system; these servos were very useful for this release system, because they had a very small footprint. The servo was approximately 1" x 1" x .5", this was ideal because it was

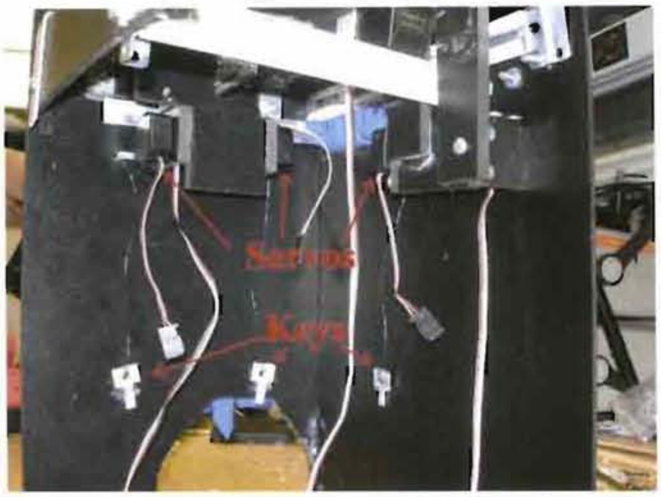


Figure 16: Storage Bins with Spandex

important that the system did not take up much space inside the robot. A voltage and control signal could be sent to the servo causing a 180° turn, pulling the key out of the paper clip, and as a result releasing the spandex and dropping the blocks into the plane. During the design and testing, we noticed that when the blocks are unloaded, they may remain in a stack sitting in the plane that causes them to come in contact with the hanging spandex as the robot drives away. However, we decided that we could make the robot drive away in a direction that allows the spandex to knock the stack of blocks over into the plane.

Processing and Sensors

Processing is a very critical and important part of the robot design since it provides the necessary tools by which the robot will be able to complete the project objectives. In the first step of processing design, a processor needed to be chosen. In choosing the proper control mechanism for this robot project, a variety of different controllers and microprocessors were examined. There were two basic types, a simple microcontroller specifically designed for robotics, such as the OOPIC, to a high-power single board computer (SBC). Factors considered in selecting an option were processing power, ability to meet the requirements of the project, power, and complexity of implementation. The current design for the robot calls for a fair amount of processing power. This complexity can be attributed to the fact that there are a lot of things that are happening at any given time, from extracting the blocks from the chute, placing them on the top belt, expanding/contracting the bins, navigating the ramp, and being able to position itself so as to deposit the blocks in the planes. At the beginning design stages of the robot many different types of processors were considered: The Basic Stamp, the OOIPC, Motorola 68HC812, PC/104 board, EPIC, EBX, Handy Board, and the Book PC.

The Basic Stamp is a simple microcontroller that was designed for basic Input and Output. Due to its simplicity, the basic stamp has become popular in robotic communities. The Stamp itself is programmed using a propriety language called PBASIC, which was developed by Parallax, Inc. This particular processor has very low processing capabilities. The most powerful Basic Stamp only had 2KB of EEPROM. This seemed highly restrictive given the amount of instructions we would need to complete the project. Due to this fact, we decided against using the Basic Stamp as our processor.

The OOPIC, or Object-Oriented PIC, is very similar to the Basic Stamp, but uses object-oriented programming to build objects that can be linked together in a visual environment. For more control over robotic devices, the OOPIC offers the ability to write custom scripts in C or Java. Even though the programming environment of the OOPIC was attractive, it suffered from the same limitations as the Basic Stamp in EEPROM. So we also ruled this processor out as well.

The next processor we looked at was the widely used embedded processor known as the Motorola 68HC812. This particular processor has 4k of EEPROM for program storage, 8MHz clock, 1k of RAM for data, 8 channels of 10-bit Analog to Digital ports, two channels of SCI, and 90+ pins of Digital Input and Output pins. While this does not offer the best in terms of program storage and speed, it has many pins available for use of input and output. This processor solves the issues of the previous considerations of not having enough input and output capability for what we would need; the memory and program storage could be a serious choke point if used solely to run the entire program for the robot. As we will discuss later, due to its low-cost, the 6812 was a good choice for use in conjunction with our main processor to poll sensors and output to our h-bridge.

The PC/104 is a form factor single board computer. This board contains the full power of a PC in a small 3.5" x 3.7" board. These can be loaded with an OS of our choice, plus there is a large availability of expansion boards that allow any number of digital or analog devices to be integrated with the main system. RAM was plentiful and ranged from 128 megabytes to 1 gigabyte. The majority of different types of PC/104 boards contained certain similarities. In specific, they have built-in USB ports, two serial ports, onboard LAN, IDE port for peripherals such as a hard drive, and a powerful processor. Due to the ability of installing an operating system such as Windows XP, one can program in any type of language he or she wishes. This means that each one of us will be very familiar with the programming environment, which will in turn help us to work more efficiently and accurately. Our robot design uses stepper motors and servo controllers that use both a serial interface and a USB interface. Using the PC/104's connection abilities, the integration and operation of these two essential devices would allow for precise control and seamless integration. Due to this integration, along with other valuable features mentioned above, this was the processor of choice in the design of our robot. We purchased the MOPSlcdVE from Kontron due to it having a starter kit that included a power supply and a development kit which turned the PC/104 into a fully functional motherboard. Information on this product can be found on our website.

In short, there was primarily one major drawback to the PC/104: the board does not come stocked with any digital input and output ports. These ports are crucial to the operation of the devices within the robot. There was however an expansion kit which added 48 input and output ports by interfacing with the PC/104 via the bus. Unfortunately, this option was far too expensive and simply not justifiable. We then chose to purchase a Motorola 6812 to use as our input and output board which would seamlessly integrate with our PC/104 through serial communication. This was a very inexpensive way to add all of the necessary input and output ports to our PC/104.

After the choice of processor and input and output integration was made, our next step was to choose the items that would control our servos and DC motors. We decided to use the Pololu USB servo controller. This controller allows 16 servos to be controlled through this one board and would connect easily via the USB port on our PC/104. Due to its high number of servo connectivity and ease of integration with the PC/104, this seemed like a very good choice. We also had an interface to operate the servos through the USB servo controller. The code, written in C++, allowed for easy integration into our development project. In fact, the other robotics team based their servo control off of the same code. By doing some research, we found that a good way to control DC motors is by use of an H-Bridge. We then decided to purchase an H-Bridge that can control two DC motors. This device accepts a 3-bit bit stream and based on these bit values, turns the motors on/off rotating them clockwise or counterclockwise. The values of the bit stream will be inputted to the device through interface of our 6812.

The navigation aspect of our robot's design was one of the key areas to be investigated from the very beginnin. Thankfully, the design of the playing board, created to the given specifications of the contest rules, very readily lends itself to line following, a well-established routine among robot designers. Using line following as our main means of navigation on the playing board was, therefore, an early consideration and almost immediately chosen. Not only is line following a fairly easy and approachable technique for navigation, but using it also meant that a large number of online resources and examples would be available. Of course the team did not discount other sources when considering the navigation portion of the design. Sensors serve as the eyes and ears of any robot. Without multiple sources of input for the processor, there are many limitations on the intelligence of the robot as a whole. Therefore, an ultrasonic ranger was used in conjunction with the line following concept as a way to make the design as robust as possible. Although the contest rules were fairly detailed, changes seemed inevitable over time. Making sure that the robot was robust meant having it handle all specifications already set in place as well as potential alterations that might make the task more difficult.

After recognizing these two main forms of navigation, individual sensors had to be chosen. For line following, simple photoresistors were investigated first. Photoresistors work like regular resistors except their value is variable like a potentiometer. The change in the light level that the photoresistor is exposed to is what triggers this variance. So, by connecting one in series with a fixed value resistor, a variable voltage could be measured and sent to a microcontroller's analog-to-digital system. The only remaining piece would be to provide a constant light source on the robot, such as from an LED, by which to characterize each individual photoresistor's sensitivity levels. Unfortunately, testing of the proposed apparatus did not prove reliable given the amount of ambient light. So, after a little research of light sensors which were implemented on other designs, the QRD1114 reflective IR sensor was chosen. This inexpensive sensor contains both an IR emitter and detector housed together along with built-in protection from outside light sources. Also, for the ultrasonic ranger, the SRF05 was purchased. This sensor was also chosen for its reliable use in other applications as well as its inexpensive price point.

After our hardware was decided upon we then needed to analyze the project description and decide on an overall program flow. We designed a program flow chart that would entail all the movements of the robot in order to complete all objectives from start to finish. Please reference **Figure 17** on the next page for the complete program flow chart.

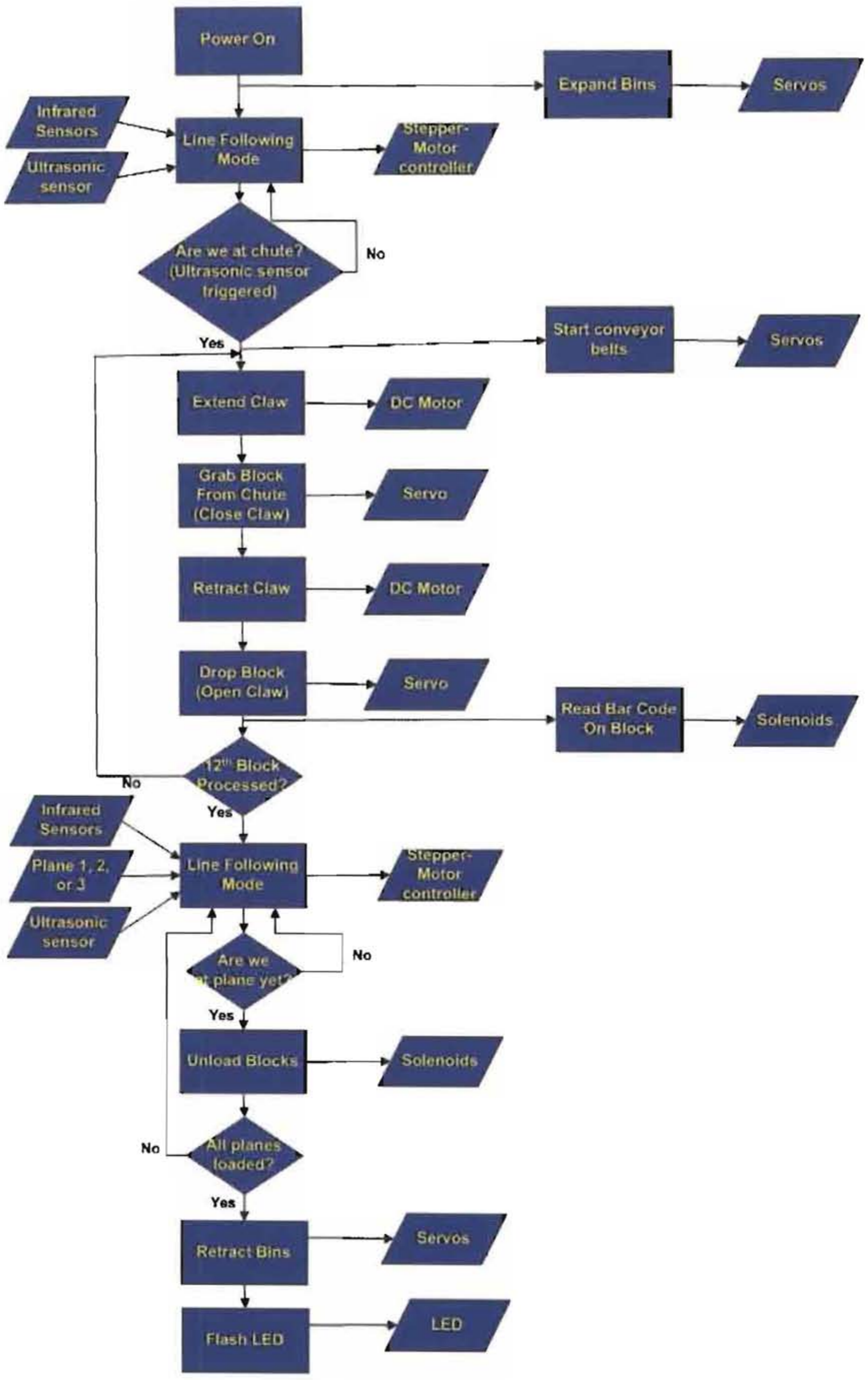


Figure 17. Flow Chart

After the overall program flow was discussed and designed, a programming language needed to be decided upon for both of our processors. With the PC/104 we were able to choose any language that we were most familiar with. This language of course was C/C++. The 6812, however, does not offer as many alternatives. Many example programs for the 6812 are written in straight assembly language, the lowest level of coding that provides the most control and optimization. Programming in assembly, however, is a long and daunting process so we began to research compilers that would compile a higher level language into 6812 assembly. The ability to write in a higher level language such as C/C++ and then compile that source code into 6812 assembly would greatly increase our efficiency and accuracy during the programming process.

After doing some research, we came across two options for programming in the 6812 environment. The first was the use of a compiler called Imagecraft. Imagecraft offered the ability to program the 6812 in C/C++ then compile that code down to 6812 assembly code. This was very appealing to us since we were very familiar with C/C++. However, with any device, there are still certain interface commands that one needs to know in order to control or access the features within the processor. Another point of consideration for the Imagecraft compiler was its rather expensive, $200, price tag. While we were doing research on compilers, we came across a language/compiler known as SBASIC. SBASIC was created by Karl Lunt who wanted to create a language that was easy to use and easy to interface with the 6811/6812. In addition to simple commands and interface, SBASIC allows for simple register control, easy trigger pulse generation, and has a simplified serial communication interface. These features, along with many resources to programming examples, caused us to choose this language/complier over the Imagecraft option.

Once the programming environment/language was chosen, code writing needed to begin. Within programming, we wanted to be as modular as possible. Modularity allows for easy troubleshooting and readability of code. We broke the code into two major groups: 6812 and PC/104. We further broke the 6812 code into three main modules, IR sensing, ultrasonic sensing, and H-bridge control. The 6812 would contain the code to interface with the sensors and H-bridge. The PC/104 would be the bulk of our code. This would be the code to process all the information received from the 6812 and also interface with the motors and servos.

Each 6812 module was written completely separate and tested. After individual modules were satisfactorily tested, we integrated the modules and tested again. We began the 6812 programming by writing the code to interface with the IR sensors. The IR sensors would connect to the Analog-to-Digital Input ports of the 6812. The 6812 would need to grab this data and send it to the PC/104 to be processed. The 6812 communicates with the PC/104 by serial communication. Keeping with our modularity scheme, we wanted to have the 6812 view the incoming data from the IR sensors and send a case based on which IR sensors saw black and which IR sensors saw white to the PC/104. The PC/104 would then accept this data and process a decision based on that case. We originally wanted to use only three sensors: one middle sensor and two side sensors. The middle sensor would make sure it was seeing white and the two side sensors would make sure they saw black. If the two side sensors saw white, then you knew that the robot was at a cross section or needed to adjust its wheels and realign.

After some discussion and thought, we decided that we wanted to have two middle sensors (left middle and right middle) and two side sensors. The two middle sensors would add a higher degree of resolution and would allow us to know if the robot was off track more quickly. This allowed smaller and quicker overall adjustments of robot direction. This would keep the robot straighter and keep it from swerving side to side. The two middle sensors are positioned 0.75 inches from each other. Since the line we are following is 1 inch wide, we only have 0.25 inches of variability in our robots lateral movement. Please see **Figure 18** for the diagram.

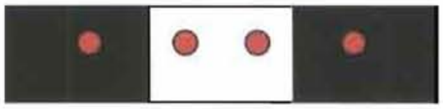


Figure 18 Sensor Layout

Next we wanted to write the code that would interface with the Ultrasonic Sensor. The Ultrasonic is used to tell us when we have arrived at either the chute or one of the three planes. This device requires one input and one output port from the 6812. The Ultrasonic sensor sends data based on distance to the 6812. The 6812 accepts this data and if the object is closer than a certain threshold, sends a signal to the PC/104. The PC/104 then receives this signal and instructs the robot to stop moving. In writing the ultrasonic sensor code, we had to become comfortable with the timing mechanisms of the 6812. We used the 6812's timer module to produce a trigger of 10 microseconds. This trigger causes the ultrasonic sensor to send an 8-cycle sonic burst. This sonic burst travels at a rate close to 0.9 ft/sec. When this sonic burst is received by the sensor, the echo line goes low. The 6812 measures the time from the falling edge of the trigger to the falling edge of the echo line. The time difference can then be used to calculate the distance to the impending object.

After the code was written to interface with our input sensors, we wanted to write the last of the 6812 source code, the interface to the H-bridge in order to control the DC motors of the claw. As stated before, the H-bridge needs to be connected to 3 digital output pins of our processor in order to receive the 3-bit bit stream (Enable bit, A+ bit, A- bit). This bit stream will be accepted by the H-bridge and based on the values (high or low) of the 3-bits, turn the motors on/off and in clockwise or counterclockwise rotation. The need for the 3 output pins is the reason that the H-bridge interface code was implemented with the 6812. The implementation of the H-bridge code was fairly simple. When we are extracting the claw we send a bit pattern with the enable bit high and A+ and A- bits low and high respectively. In order to retract the claw we send the same bit patter with A+ and A- bits swapped. The H-bridge code communicates with the PC/104 over a serial connection in order to know exactly when to extract and retract the claw.

Next was the task of writing the source code for the PC/104. The PC/104 is used to receive the sensor data from the 6812 and, based on that data, send movement/control commands to the robot. The code for the PC/104 is broken into three parts: line following, servo control, and stepper motor control. The line following code is able to receive the case sent by the 6812 (based on IR sensor status) and based on that case, send ASCII movement commands to the stepper motor controller. These commands are sent to the stepper motor controller by serial communication. The PC/104 controls the servos by interfacing with the servo controller via USB. The servo controller has a number of functions available to send commands to the servo motors. In our implementation we used the absolute position command. Servos operate based on a pulse signal, typically ranging between 1 ms and 2 ms, refreshed at 50 Hz. The servo controller's circuitry generates this 50 Hz pulse so that all we have to do is tell the servo controller what position we want the servo at. The controller accepts values ranging from 500 to 5500. These values are found by multiplying the desired pulse width by 2000. Based on the specific needs of each servo, we calibrate the motors in order to find the correct pulse widths for their respective operations.

Once each modular piece of code had been written to control/interface with each device, the code needed to be combined and program flow incorporated. The PC/104 controls the overall flow of code and which part of the code needs to be implemented when. There are six parts of the code that need to be executed representing the six modes the robot needs to be in during the duration of the performance: line following, bin extend/retract, bin loading/unloading, extend/retract claw, open/close claw, and flash LED. The robot begins by powering on. Once powered up, the robot needs to enter line following mode. The PC/104 sends a message to the 6812 telling it that it is ready to receive data from the IR sensors for line following. The PC/104 receives this data and sends movements commands to the stepper motor controller. At the same time, the PC/104 sends a servo command to the servo controller to extract the bins. From this point on, the hardware timer for the 6812 continues generating trigger pulses to receive a distance measurement from the ultrasonic sensor and stop the robot in front of the chute when a particular distance is satisfied. This ends the first part of the flowchart's three major processing stages. The next stage involves a lot of handshaking between the 6812 and the PC/104. First, the 6812 sends a command to let the PC/104 know that the motors must be stopped. Upon confirmation, the 6812 uses the H-bridge to control the dc motor and extend the claw. Once completed, the 6812 sends another command that allows the PC/104 to operate the claw servo and grip a block. Finally, the 6812 receives confirmation and retracts the claw with the H-bridge and dc motor. At this point, the PC/104 needs only operate the servo to release the block onto the vertical lift system and determine whether or not all twelve packages have been extracted. If all twelve packages have not been extracted, then the process continues. Otherwise, the robot goes back into line following mode. There is a new input variable in this line following mode, destination plane. Decisions on which direction to take at an intersection are based on the current destination plane. Once the destination plane is reached, determined by the ultrasonic sensor, the PC/104 instructs the servo release mechanism on the plane's bin to open. The packages are released into the plane, and the robot continues back into line following mode until all planes are reached. After all planes have been loaded, the PC/104 instructs the robot to retract its bins. Once the bins have been retracted, the robot flashes a blue LED signaling completion of the round.

Conclusion

The robot was constructed and programmed enough to qualify for competition, meaning that it can drive forward following a line and extract a block from the package chute. Due to hardware problems, we were unable to complete the whole robot. However, each individual system performs as expected, and the entire system just needs a new processor to be fully functional.

Improvements that could be made include speeding up the extraction procedure. This could be done by increasing the speed of the claw, the vertical lift system, and the horizontal belt system. Another way to improve the design would be to provide more sensors for line following. This would provide better precision and less wasted side-to-side movement while the robot follows the line. These improvements would make the robot much more competitive in that they would reduce the time it takes to complete the course.

Overall, the design of the robot is solid and will be competitive in competition. By changing some programming and speeding up some motors, the speed could be greatly increased. By altering some sensors, precision can be gained and gain speed even more. With these improvements, the robot could be greatly enhanced.

Appendix A: Track Layout