# Vex™ for the Technically Challenged

(by the Technically Challenged)

Version 2.0

## Table of Contents

# Introduction

To date, relatively little documentation is available on the Vex™ robotics system, and much of what is available requires a prior level of robotics experience. This guide is intended for those who are starting with Vex™ from ground zero or are moving up from a simpler system like LEGO™ Mindstorms.

You may find that much of the information here is painfully simplistic and obvious. It was collected as the result of many "duh" moments which were the culmination of much time ~~wasted~~ invested over the past year, time that we hope to save other newbies like us. Our team participated in the *FIRST* Vex™ Challenge pilot competition in April 2005, but with a non-engineering coach and largely self-taught students, we found it difficult to learn and improve following the competition, due to the lack of training materials. Cool designs on the Innovation First, Inc. website[1] and Chief Delphi forum[2] provided tantalizing building ideas, but we were unable to reverse engineer many of these designs without an overall understanding of basic engineering principles. With no *FIRST* teams within a 90-mile radius, and no engineers within our circle of acquaintances (except one who drives a train), we were left to trolling the internet for assistance and ideas. We made it our mission to collect and organize these ideas for the benefit of other startup teams (and ourselves). In many cases, what we present is not the optimal solution, but simply something that gets the job done, by hook or by crook, leaving the best challenges for you to solve. Many engineering design concepts were not mentioned because working models were not easy to construct using Vex™ parts.

We would like to thank Art Dutra IV, Kathie Kentfield and the Chief Delphi forum community for patiently answering our obvious questions, Keith Petersen for editing assistance, and Jim and Conni Bock for their instruction and inspiration. We have attempted to acknowledge all sources and photos, but if we inadvertently missed someone, please let us know. Unlabeled photos were taken from our personal library.

Yolande Petersen, Team Mom & Coach
Justin Petersen, Master Builder & Technical Guy

## Getting Started

Storage

After you unpack your Vex™ kit, it is advisable to purchase a storage container with compartments, like a tackle box, for the various pieces. Ideally, the compartments should be of varying sizes: some long enough for the long bars and axles, and some small enough for the screws, nuts and washers. The cardboard box which the kit comes in is useful for storing larger items like the remote, wheels, and the manual.

Photo used by permission from Art Dutra

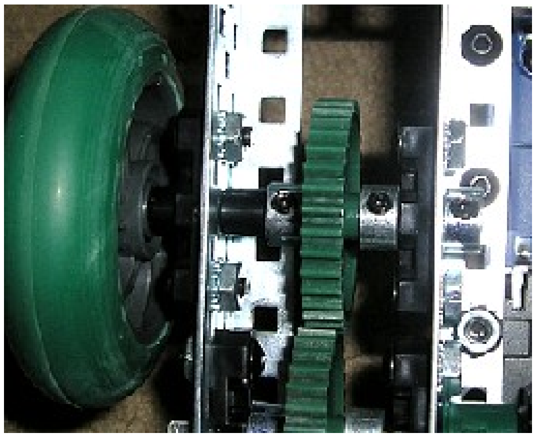
Manual (the Inventor's Guide)

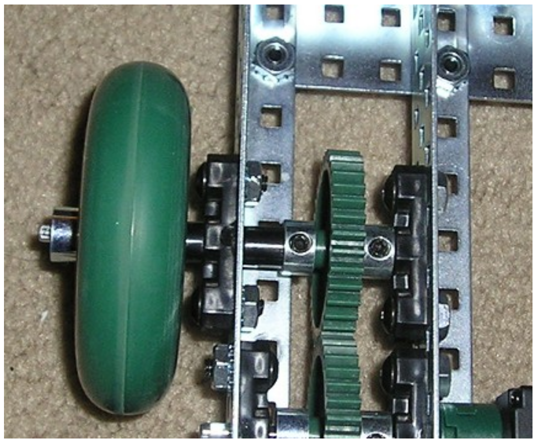
The Inventor's Guide should be filled with pages and separated into sections (some earlier versions of the kit came with a file on CD that had to be printed). Some of the sections are empty and will be filled up as you purchase additional accessories and add the documentation to your manual. It's a good idea to read through the manual, even though "nobody ever reads the manual."

Squarebot

From the Inventor's Guide, you can build the Squarebot (instructions in the Structure section, 2.2-2.26). This will help you become accustomed to the parts and tools in the set, as well as giving you a robot that actually works. Try driving the robot using the remote control on a variety of terrain to test the limits of your robot's speed and power.

One design improvement is suggested here. Notice that the Squarebot wheels are held onto the axles by friction, and that they tend to fall off. This problem becomes exacerbated with repeated use. Securing a collar to the outside of the axle solves the problem; however, if the Squarebot is built as shown, there is insufficient axle for this purpose. Removing the innermost collar allows the axle to be slid outward, and proper positioning of the remaining collars against the rails prevents sliding of the axle back and forth. Also, on p. 2.20, the use of spacers is suggested if the gears slide; these are generally required.


Original Squarebot axle, collar on inside


Modified Squarebot axle, collar on outside

## Modification of Parts and Additional Tools

One thing which distinguishes the Vex™ system from many other robot kits is that its pieces are meant to be modified through bending and cutting. While it is wise to be conservative about cutting, a balance needs to be found between settling for an inferior design to avoid cutting, and wasting materials. Keep all cut pieces, no matter how small! They will almost always be reused in some way.

Having some pieces pre-cut will make your project go faster, an important consideration if you have a team of impatient students. There are some sizes of pieces that will likely be used if you use your Vex kit extensively. See Appendix A for cutting suggestions.

Minimal cutting tools required include tin snips, a hacksaw for angle bars (including a vise for securing pieces), and a file for removing burrs. More expensive power tools can also be used. Appendix B contains a list of additional suggested tools.

Another helpful item is a small platform for elevating the center of your robot, keeping the wheels off the ground. This enables you to download programs or perform tests with the wheels spinning without having the robot run around the room, tangling the cords. A stack of paperback books or 2" X 4" blocks does the job.

## Upgrades and Add-ons

While it's sad to think that you can barely get your kit unpacked before thinking about spending more money, there are a few additional purchases you should consider that will add to the enjoyment of your kit. The first is the Power Pack, which will save money in the long run over AA batteries, and is the only power source permitted in some competitions. It's also smaller and lighter than AA batteries and will give you more flexibility as you design your robot. The second is the Metal and Hardware Kit, which will provide replacement parts for the ones you ~~mutilate~~ modify. Finally, the Programming Kit is a necessity if you want the robot to be autonomous (self-controlled) rather than remote-controlled, and it is required for sensor use. One way to obtain the basic add-ons cheaply is to purchase the bundle pack offered by Innovation First, Inc. which is offered to teams registered for the *FIRST* Vex™ Challenge. More commentary on additional add-ons can be found in Appendix C.

Vex™ accessory packs often have "goodies" hidden in the packaging. Be sure to completely empty the box and check for documentation. Unfold/unwind all cardboard packaging, and cut or pull off any large pieces of packing tape to be sure that you are not discarding small hidden treasures. Better yet, perform an inventory of all the parts listed on the side of the box. Most accessory packs come with documentation that is intended to be stored in the Inventor's Guide binder.

## Locomotion (Getting Around)

### Gears, Speed, and Strength

You may have observed that your Squarebot is relatively fast and can navigate fairly rough terrain. In case you're wondering, "Can I make my robot go faster?" and "Can I make it stronger?" the answer is "yes" to both, but you can't do both at the same time.

Take a look at the gear train on your Squarebot. The motor is connected to a 60-tooth gear, while the wheel axles are connected to 36-tooth gears. Each time the motor makes one complete rotation, it moves the large gear through 60 teeth, which in turn, also moves the smaller gear through 60 teeth. However, 60 teeth on the 36-tooth gear represent nearly 2 rotations (one and 24/36 rotations, to be precise), which means that the wheels have a faster turning speed, or *angular velocity,* than the motor. You have just found a way to make your robot go faster! This process of powering a larger gear to make a smaller gear go faster is called *gearing up* (think of a race car).

While increasing speed can be a good thing, the trade-off is a reduction in strength, or *torque*, and at times, it is desirable to reduce speed and increase torque by *gearing down* (think of a tank). This is accomplished by powering a small gear and using it to drive a larger gear.

Ground speed, or *linear velocity,* can also be modified by changing the wheel size. If the motors have a fixed number of rotations per minute, using larger wheels (with a larger circumference) will enable the robot to cover more distance with each rotation.

Calculating the gear ratio (drive: driven) can give a precise calculation of how much speed (or torque) is gained. In the case of the Squarebot, the ratio is 60:36, or reduced, 5:3. We can obtain a ratio of 5:1 by using the 60-tooth gear to drive the 12-tooth gear, and even higher ratios can be obtained by using more than 2 gears in a gear train. One caution is that some torque is always lost to friction with each conversion, so minimizing the number of gears in a gear train is generally desirable (even though the longer trains look more impressive). More on gearing can be found in the Structure subsection of the Inventor's Guide.

### Driving Base Considerations[3]

The driving base is the foundation of all other aspects of robot design. Optimally, you want to find the right balance of speed, strength, stability, and maneuverability to accomplish the desired task. Obviously, the optimal base needed for driving at maximum speed on a smooth, flat surface will differ from one needed to climb up a 20-ft. garbage pile. As described above, the balance between speed and strength is accomplished by adjusting the gear ratios between the input (motors) and the output (wheels).

*Maneuverability* can be defined as the speed and precision with which the robot can change direction. While a high degree of maneuverability can be desirable, it can actually be a disadvantage at times, as a robot which changes direction too easily can be difficult to control.

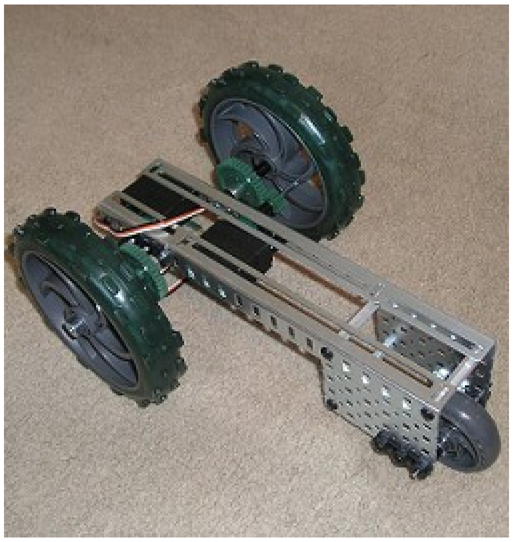
### Two-Wheel Drive

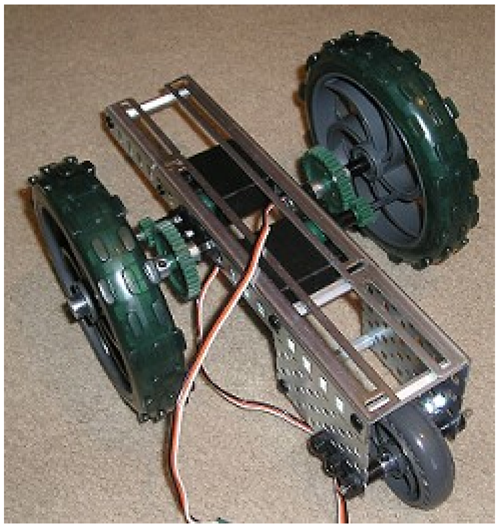
The *two-wheel drive*, or *castorbot*, is highly maneuverable and relatively easy to build. It requires only two drive wheels (one on each side) and support of the chassis to keep the robot from tipping over. Supporting the chassis with two free-spinning wheels is possible, but the friction on the unpowered wheels often makes turning difficult. Removing the rubber from the "dead" wheels greatly reduces the friction. Alternatively, a castor may be used to support the chassis. However, a traditional pivoting castor (like on a swivel chair) is often unpredictable, as its starting position affects the direction of motion. A better choice is an omni wheel or a skid plate (a low-friction object which simply drags on the ground). Creating a skid plate from an object with a rounded surface, like a rubber-less wheel or the Vex™ bearing block, is preferable to objects with corners, which snag easily.

There are several disadvantages to the castorbot. It turns with relative ease because there is virtually no side friction working against the wheels of the robot as it turns. However, with no friction regulating the robot's speed of turning, its inertia will always make it "want" to continue turning even after the motors have stopped. In addition, the unpowered wheels or castors have no pushing power. Finally, castorbots are very difficult to drive in reverse.

There are ways to offset or compensate for these disadvantages. When designing a two-wheel drive train, the powered wheels should be close to the center of gravity. This allows the robot's pivot point to remain close to its center of mass, minimizing the area through which it must travel in order to turn.

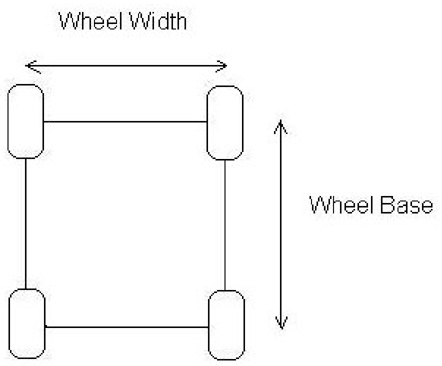
Below are 2 configurations of a simple castorbot (controller & battery pack removed for better view of gearing). It is geared down to compensate for the large wheels and turns easily. However, one design flaw should be noted, the result of a tightwaddish reluctance to cut 2 additional rails. Note that the axles are cantilevered; that is, they are supported on only one side of the gearing. This is an unstable design because plastic bearing blocks are bendable and do not perfectly align with the metal. If too much weight is applied to the center, the axles bend inward, producing wobbly motion. A better design would be to insert another set of rails between the gears and the wheels, supporting the gear train on both sides.


Drive wheels far from the center of gravity (CG), with significant drag from the castor.

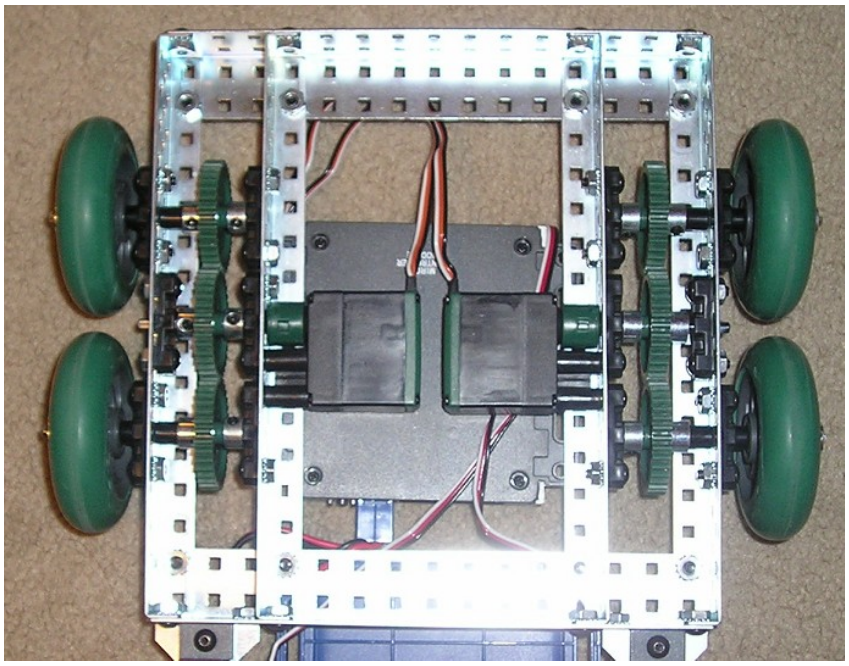

Drive wheels closer to CG, less drag.

Four-Wheel Drive

Another type of drive train is the *four-wheel-drive* system; the Squarebot is an example of this. With the addition of two extra driven wheels, a four-wheel-drive robot has more traction and control than a two-wheel-drive robot. The trade-off is the increased wheel base (distance from front to back wheels), which can cause problems turning, especially when the wheel base exceeds the wheel width (distance from left-side to right-side wheels). On the other hand, the longer wheel base adds stability – the robot is less likely to tip when stopping suddenly. It is thus more capable than a castorbot of sporting large attachments, especially ones with a high center of gravity.

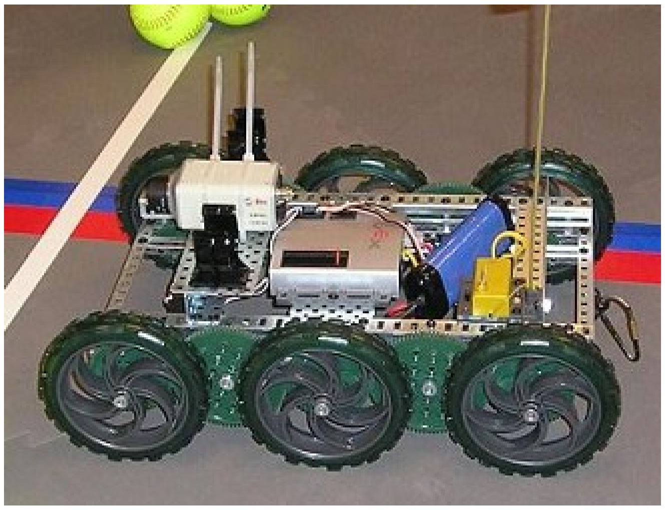
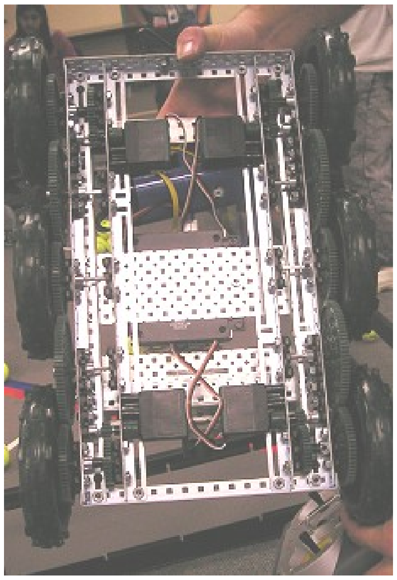


Another 4-wheel drive example is the Slowerbot (pictured below, upside down), geared with a 1:1 ratio.



## Six-Wheel Drive

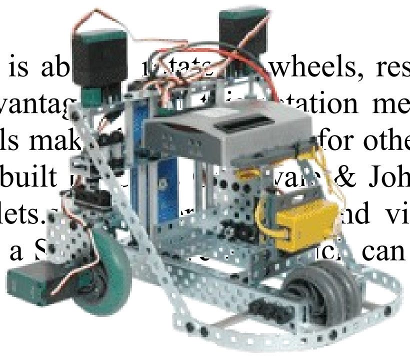
The *six-wheel-drive* robot offers a good compromise between traction and maneuverability. Commonly, the center wheels are approximately 1/8" - 3/16" lower than the front and back wheels. Thus, the robot rests on only 4 of its 6 wheels at any given time, so the effective wheel base is only the distance from the center wheels to the front or back, or about half the distance from front wheels to back wheels. This shortened base allows the robot to turn more easily, while greater total chassis length provides more stability.

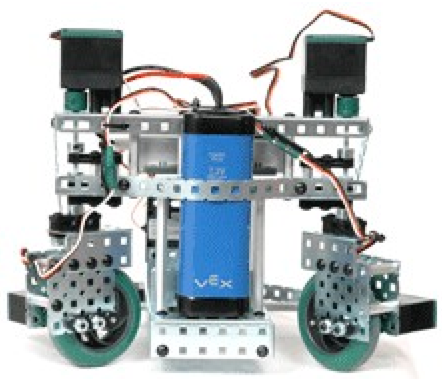



Photos used by permission from Mark Edelman

## Swerve Drive

A *swerve drive* (aka crab drive) robot is ab [illegible] wheels, resulting in a great increase in maneuverability. The disadvantag [illegible] tation mechanism is more complex to build, and the pivoting wheels mak [illegible] for other robots to push around. An example is the Swervebot, built [illegible] & John V-Neun, which can be found at vexlabs.com/vex-triplets. [illegible] nd video of the Swervebot available here). Vexlabs also sells a S [illegible] can be used as an add-on to the Starter Kit.





Photos used by permission from Innovation First, Inc. Photos or materials may not be copied/reproduced without permission from IFI.

## Holonomic Drive

Vex™ omni wheels (purchased separately) make it possible to construct a *holonomic drive,* or *omni drive.* A holonomic drive enables a robot to rotate in place, move in any direction, or do both at the same time. The major advantage is a great increase in maneuverability without having to add an entirely new mechanism to turn the wheels – a holonomic drive can change direction without changing wheel position. The major disadvantage is that the robot isn't very good when it comes to pushing and shoving. Omni wheels tend to have poor traction, as they are generally made from hard, low-friction material without treads. Also, the wheels are controlled independently, adding complexity.

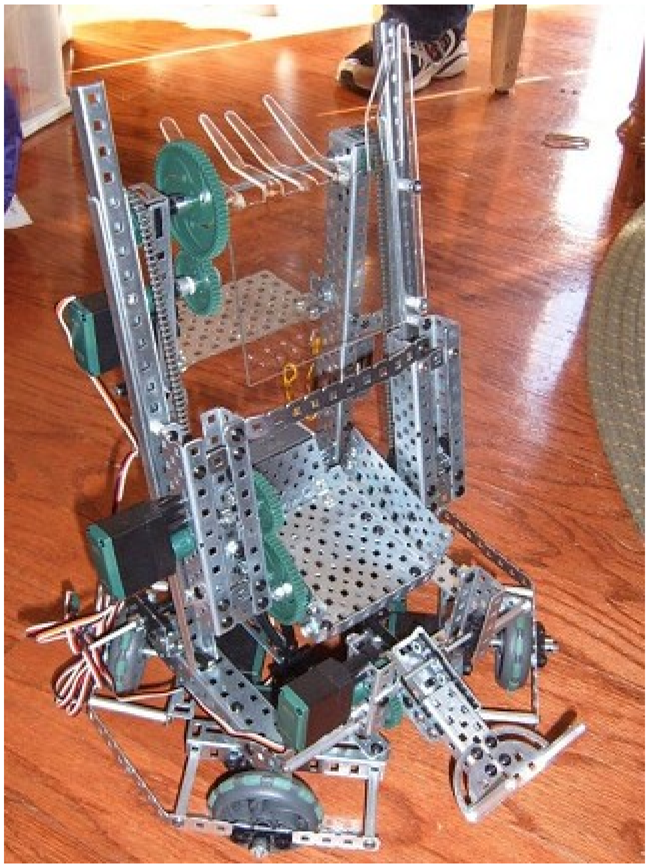


Photo used by permission from Art Dutra

## Chassis Layout

The geometry of the chassis is also an important consideration, and a number of configurations are pictured in the 1st row of the diagram below, taken from Team Unlimited's PowerPoint[4]. Chasses are commonly rectangular, and the ratio of wheel base to wheel width can be adjusted. A short wheel base & wide body (pictured in the upper left) are better for turns and side stability, while a long wheelbase & narrow body (2nd from left) are better for front to back stability. A square chassis is a compromise for both. U-shaped or H-shaped chasses are more fragile than rectangular chasses, but are useful when trying to stay within a size limit while making room for a manipulator which touches the ground, as demonstrated by the 3rd object in the 2nd row of the diagram. While the wheel axles are typically attached to the chassis (1st object, 2nd row), the chassis may ride above the wheels (2nd object, 2nd row).

## Sample Vex chassis configurations

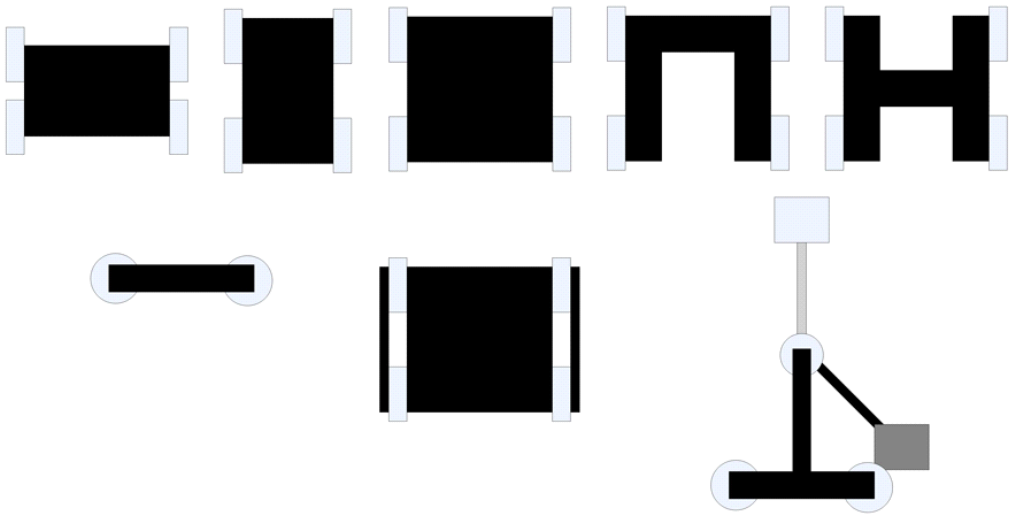
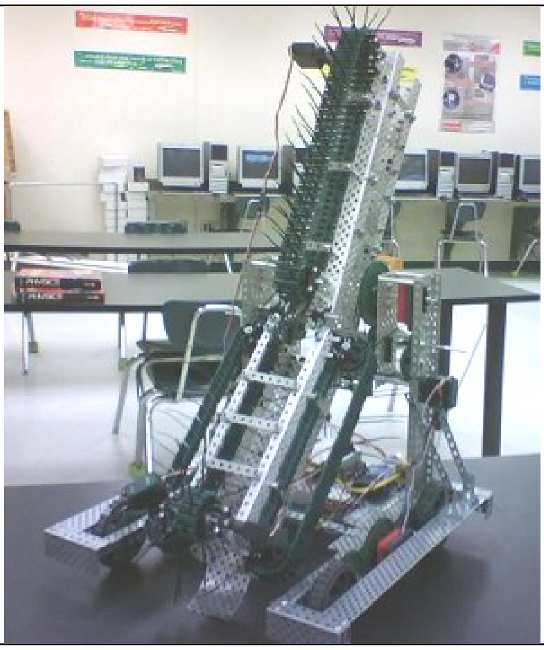


Diagram used by permission from FVC 2013 ('06-'07) Team Unlimited.

Pictured below is a U-shaped chassis. Note that the absence of a bar across the front allows ample room for the ball collector.



Scorpio; photo used by permission from FVC 3220 ('06–'07) S.P.A.M.

## Manipulation of Objects (Gripping and Lifting)[5]

One of the important applications of robotics is the ability to move objects. Handling dangerous objects (like land mines) and carrying heavy objects (like food supplies and homework) are just a few of the things that robots are designed for. Two important aspects of manipulation are gripping (grabbing) and lifting.

### Grippers

Grippers are used to pick up and move objects from place to place. There are many types, a few of which are described below.

Linear One-Axis Grip (pinchers)

Imagine grabbing an object, like a soda can, with only your thumb and forefinger. You would probably attempt to grab the can by either 1) pinching it with the tips of your fingers or 2) encircling it with the entire length of the arc formed by your thumb and forefinger. In the same way, a linear one-axis grip holds an object with either the tips of its arms or by encircling it with its arms, the way you would hold onto a very large beach ball. For round objects, the second method requires less force to hold an object, as the greater contact surface provides more friction. The first method also requires that the "finger tips" be positioned very precisely, but is better for picking up small, light objects of irregular shape.

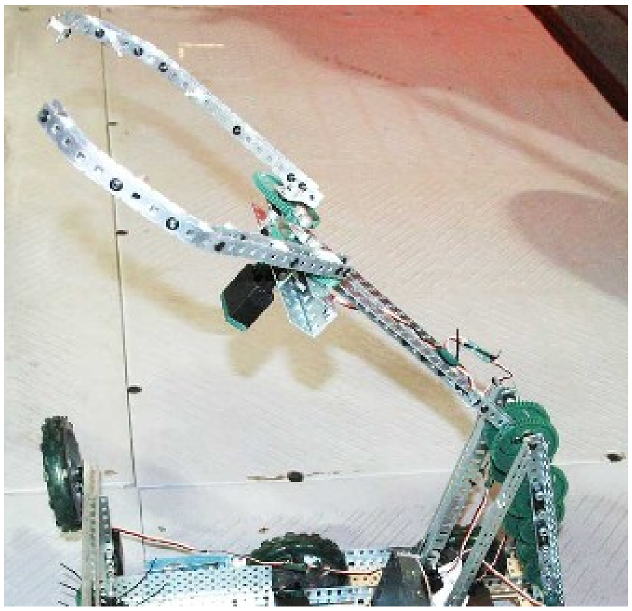


Photo used by permission from Innovation First, Inc. Photos or materials may not be copied/reproduced without permission from IFI. Team #53 ('05 Pilot), if you would like to be acknowledged by name, please contact us.

Linear Two-Axis Grip (claws)

Now envision grabbing an object with 3 fingers – this is how a two-axis grip works. The advantage here is that you don't have to pinch the object or hold it solely by friction, as the object can be nestled within the fingers of the claw without being squeezed by them.

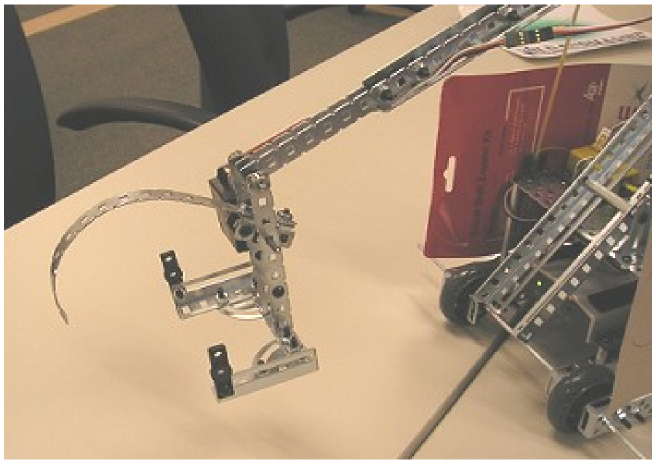


Photo used by permission from Gary Yeap & FVC 3614 ('06-'07)
TBA (Tee-Ba)

Roller Grip

A computer printer with roller feed uses a roller grip – the spinning rollers on either side use friction to move the paper into place. In theory, by reversing the direction of the rollers, you could spit the paper back into the paper tray. Depending on the size and shape of the object, a roller grip can have very good holding power. The Vex™ kit does not contain actual rollers, but various mechanisms using this principle can be constructed. For example, S.P.A.M.'s robot (pictured in the Chassis section) uses a conveyor mechanism for gripping balls. Another example is the robot pictured below, which whose rollers function like the beater brush on a vacuum cleaner.

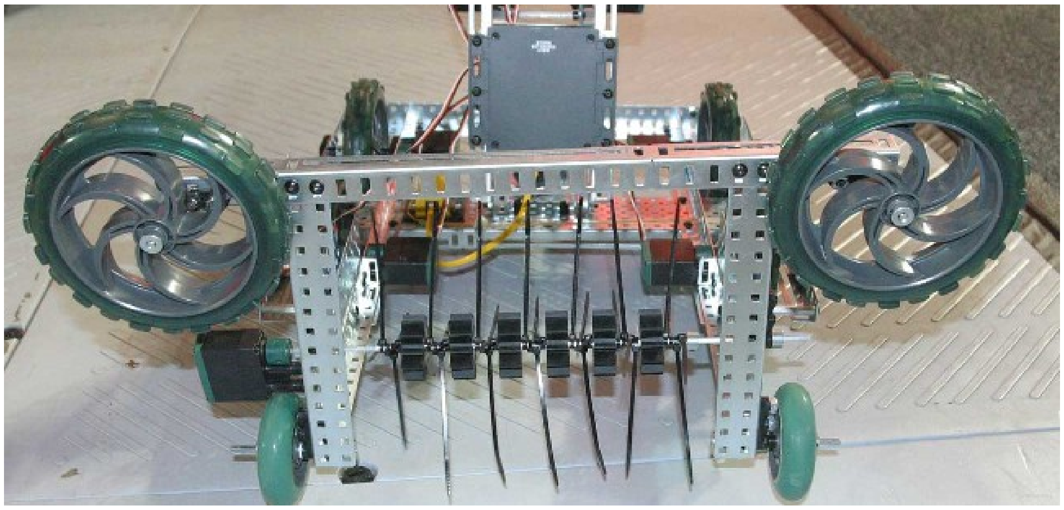
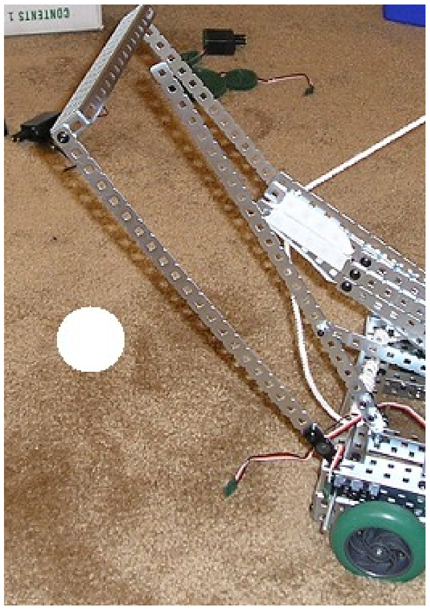
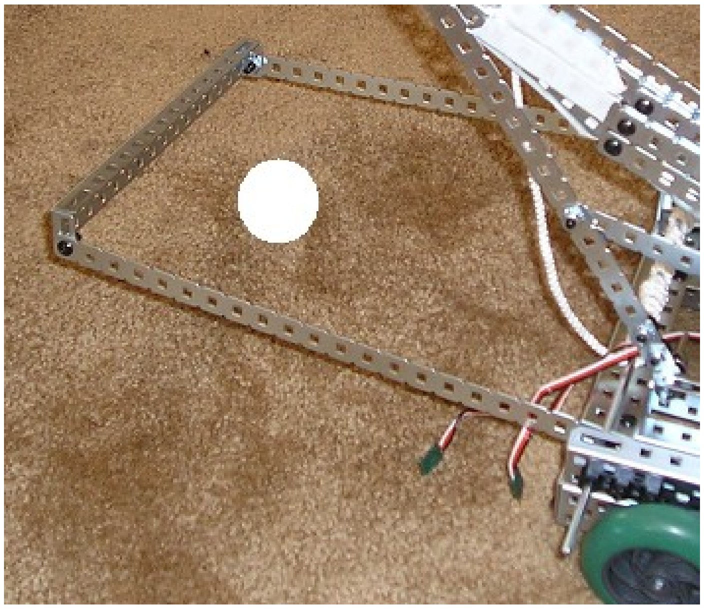


Photo used by permission from Innovation First, Inc. Photos or materials may not be copied/reproduced without permission from IFI. Team #53 ('05 Pilot), if you would like to be acknowledged by name, please contact us.

## Hoop Grip

A hoop grip traps objects by surrounding an object with a hoop (covered or uncovered), then releasing it by lifting the hoop (or dragging the object and allowing it to drop to the ground). While simple to construct, it tends to be unreliable and is not generally recommended.


"Ball" released


"Ball" trapped

## Choosing an Optimal Gripper

There are a number of factors to consider in designing grippers. First, the size, shape, and weight of the object to be picked up will affect which type of grip is optimal. Whether the object is on or off the ground and where it needs to be moved will also play a role.

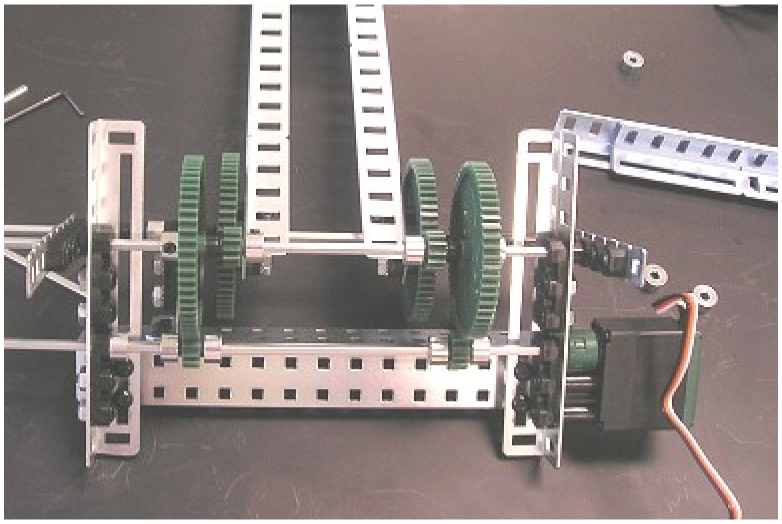
Linear grippers are versatile; that is, they can pick up objects of a variety of sizes and shapes, and carry objects to and from a variety of locations -- objects do not need to be dragged or rolled along the ground. They do require that the arm be precisely placed in relation to the object to be grabbed. Roller grips are much more forgiving in this way, as the entire length of the roller is used for gripping. Hoop grips tend to be the least versatile, as they often require that objects be dragged or dropped, rather than carried.

Speed of grabbing and release are important considerations. A linear grip which is powered by pneumatics will operate more quickly than a motorized grip. A roller grip tends to be slow.

The ability of a gripper to "hold on" is another consideration. A pneumatic grip can hold on without being powered because of its bi-stable (open or closed) configuration. A motorized one-axis grip must be continually powered while carrying an object because continued pressure/friction is required to keep hold of the object. The disadvantages of the pneumatic grip are that the parts are expensive, and use of pneumatics is not permitted in all competitions. Roller grips hold on without continual powering.

## Arms

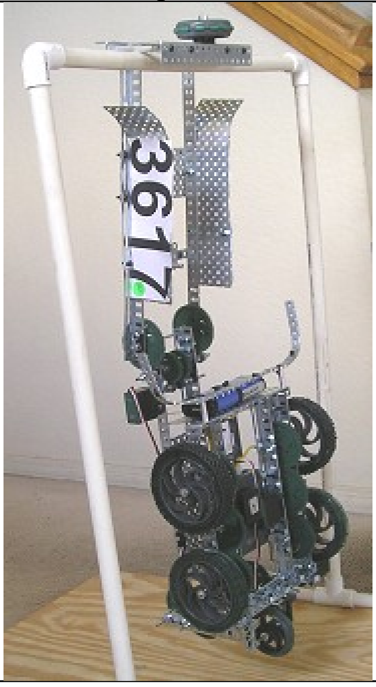
A robot may be required to lift heavy objects (including itself), and there are a number of ways to accomplish this. The simplest device is an arm with a single axle driven by a motor that rotates the arm up and down. Generally, connecting an arm directly to a motor is not desirable, because the motor has too much speed and not enough torque for most practical purposes – gearing down in 1 or 2 stages produces a better design. The arm shown below uses 2-stage reduction, with a 1:25 gear ratio (1:5 and 1:5). It can be attached to the back of a Squarebot or other robot. Note that double gearing (duplicates on right and left) adds strength and reduces the likelihood of stripping the gears.



Additional powered "joints" can be added, which should also be geared down. The robot below has a 2-jointed arm and is designed to chin itself up.

Above: Upper and lower hinges retracted.
Right: Upper hinge extended, lower hinge retracted. When both upper and lower hinges are extended, the arm grabs the bar while all 4 wheels are on the ground.

Photos used by permission from FVC 3617, Metal Gear

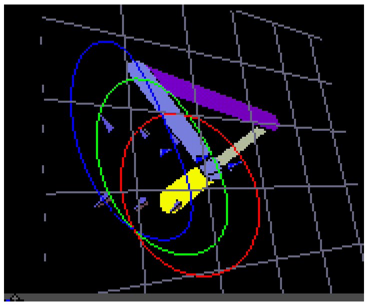


## Lifts

*Articulated lifts* use multiple jointed sections that unfold to lift a platform or bucket. An example of this is the four-bar linkage, described below. *Telescoping lifts* have multiple overlapping sections that move into and out of one another. Two examples of these are extension lifts and scissor lifts, which will be discussed in more detail.

### Four-Bar Linkage

Circular motion produced by motors can be translated into linear lifting motion. The four-bar linkage is one way to accomplish this. Check out the animated gif: www.linkagedesigner.com/res/fourbar.gif [6]



The gray bar is fixed at both ends, and as the yellow bar travels in a circular pattern, the end of the purple bar (where it is joined to the blue bar), oscillates up and down. This is similar to the motion produced by the windshield wipers of a car. A crank and piston also operate according to this principle. Below is a Vex™ example of a four-bar arm.

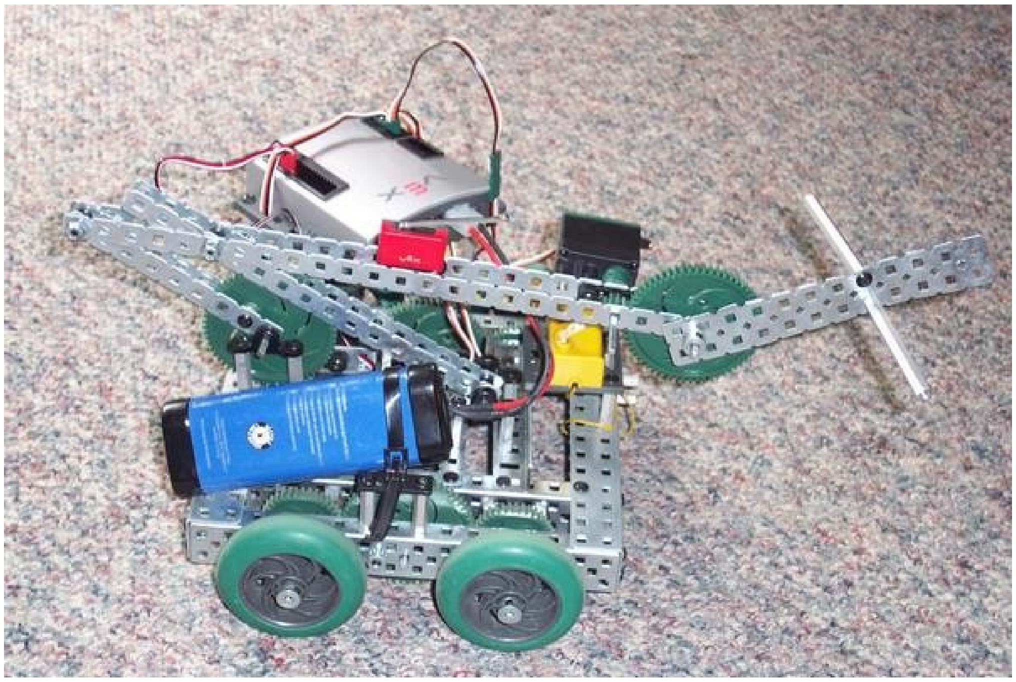
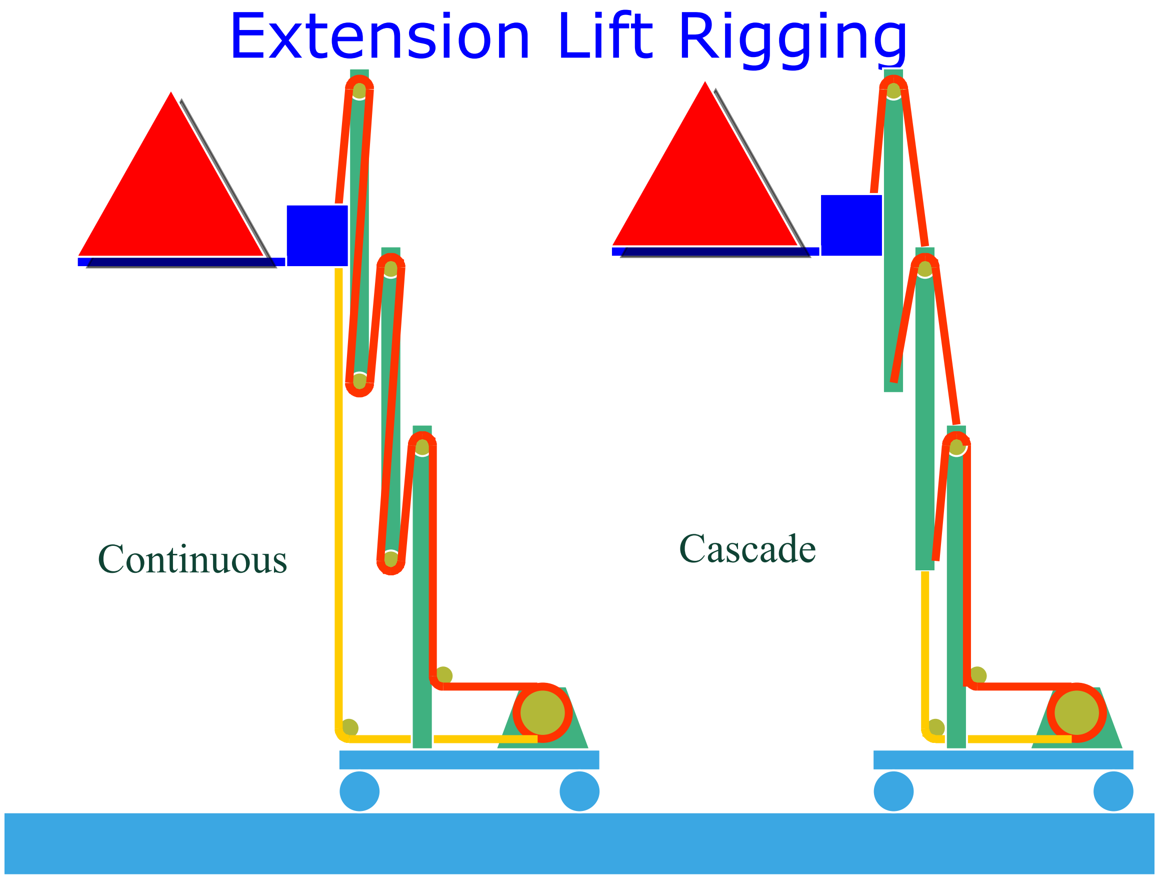


Photo used by permission from Michael Leicht

Another example of a 4-bar arm (in action) is the BottleBot at: vexlabs.com/vex-bottlebot.shtml.

Extension Lifts

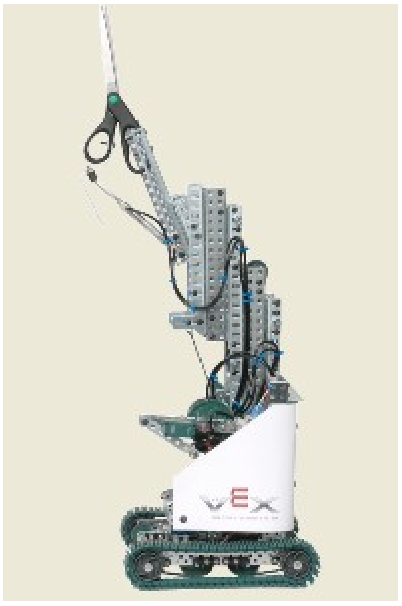
An extension lift also allows an arm to lengthen in a linear fashion by tightening a cable or chain around a drum, causing overlapping sections to extend or retract. Powering is required for both extension (going up) and retraction (going down). Greg Needel's PowerPoint[5] contains the diagram shown below:

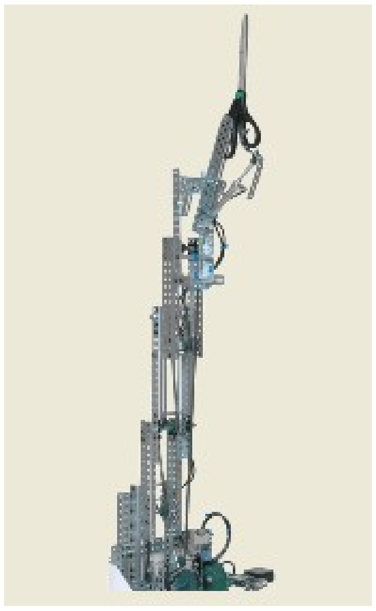


Each design has its own advantages and disadvantages. Lifts with *continuous rigging* move evenly but slowly, as more cable is required for full extension. The cable uses identical speeds for going up and going down, so only one drum is required. The disadvantages are that the intermediate sections tend to be more prone to jamming, due to the longer cable length and lower cable tension. In addition, the longer cable requires that more cable be moved in the same amount of time, requiring a faster motor.

Lifts with *cascade rigging* require that less cable be moved to fully extend, and the intermediate sections are less likely to jam. However, the cables for going up and going down use different drums requiring different speeds, and higher tension on the lower-stage cables requires higher gearing to deal with the greater force. The greater number of connections increases the complexity of the system and potential for something to go wrong.

The ribbon cutter robot below by John V-Neun and Chris Carnevale uses a 5-stage extension lift. You can view more detailed photos and a video of its operation at: vexlabs.com/vex-ribbon-cutter.shtml.


Retracted

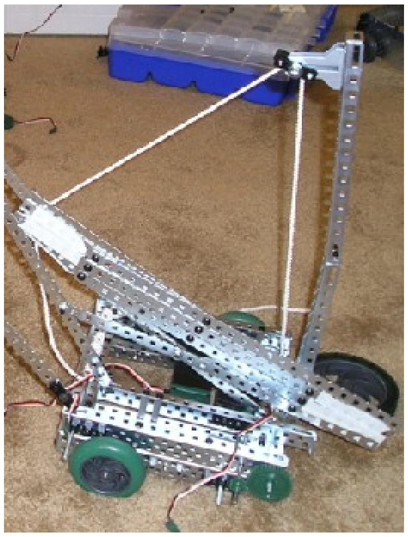

Extended

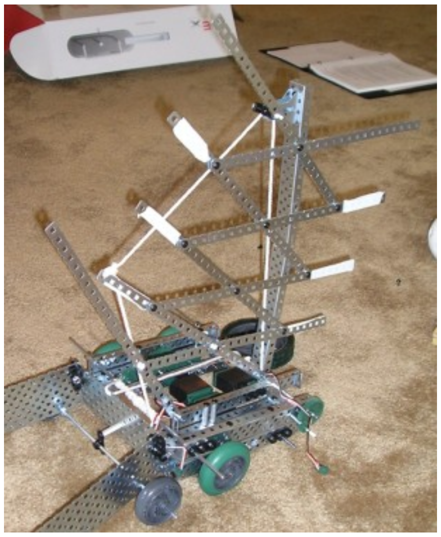
Photos used by permission from Innovation First, Inc. Photos or materials may not be copied/reproduced without permission from IFI.

## Scissor Lifts

Scissor lifts also have overlapping sections that extend in a criss-cross pattern. One advantage that they have over extension lifts is that the retracted height is minimal, enabling the lift to easily get under a load. The disadvantages are that the large number of criss-cross bars increases the weight, and that the stability decreases as height increases and the lift approaches maximum extension.

The lift below is activated by a pulley. The taped ends were eventually cut off when the desired length was found by trial and error – measure twice, cut once. The use of a single thickness of long bars was a poor design choice, as the long bars tended to bend when the robot attempted to lift itself. A better choice would have been to use a double thickness in the scissor lift, separated by spacers or short threaded beams. Although this would add to the weight of the arm, the increase in strength and stability would more than compensate.


Retracted


Extended

## Programming

The tutorial which comes with the programming kit (printed pages to be inserted into the Inventor's Guide manual) provides excellent step-by-step instruction, and it is advisable to work through it without skipping steps. The following information supplements the tutorial and presumes that you have completed it. While it is possible to use EasyC for a first programming experience, the tutorial does not teach generic programming concepts in a comprehensive way, and it would be helpful to have mastered important concepts such as inputs, outputs, looping, and conditionals in another introductory programming course.

### Remote and Autonomous Control

You can program your robot to operate by remote control as described in the tutorial by using the commands under RC Control on the menu. You can also program your robot to operate completely autonomously, with no control from the transmitter. However, certain versions of EasyC for Vex contain a glitch that requires that prevents autonomous programs from executing unless the microcontroller receives a signal from a transmitter, even if the robot is not affected by any input from the joysticks or buttons on the transmitter. If your robot refuses to function after a program is downloaded, try connecting an RF receiver (yellow box) and turning on a transmitter.

### Downloading Tips

Downloading can be a slow process, but it is highly recommended that you exercise patience and allow the process to complete itself. In at least one case, interrupting the process has resulted in a completely frozen and unusable system: not being able to run the old or the new programs, and not being able download anything new (ouch!). If you do experience a system freeze, try removing all batteries and power for a period of time (it may take an hour or more), and try again.

Be aware that your program will begin running the instant that downloading is complete, which may result in your robot making its way to Alaska before you have time to unplug the cable. One solution to this dilemma is to prop up the robot chassis on a platform (books or 2" X 4" blocks work), so the wheels are off the ground while downloading. Another is to use the "interrupt" button, located on the box on the orange download cable.

### Motors

According to the manual, motor speeds are regulated by setting the motor direction, with 255 producing the maximum clockwise (CW) speed, 0 producing the maximum counter-clockwise speed, and 127 producing a motor stop. More precisely, the motor speeds increase gradually over the following ranges:

CW rotation: 128 – 255 (128 produces a motor stop, 255 produces the greatest speed)
CCW rotation: 127 – 0 (127 produces a motor stop, 0 produces the greatest speed)

Actually, a stopped motor can be produced by a range of values, rather than a single value. To find the true range of stop values, use the Online Window, described on page 8.21 in the Programming section. Drag the slider bar through the range of values until you observe your wheels actually moving. A free-spinning axle might begin to move CW at 135, while a loaded motor might require a value of 160 to produce any CW rotation, since it has more friction to overcome. This is an important consideration when trying to program your robot to move slowly. A robot which is programmed too close to the actual stop threshold tends to stall with conditions which add friction, such as adding weight, turning, and driving on different surfaces. In general, setting motors at full power is desirable, unless there is reason to do otherwise. Speed can be reduced mechanically by gearing down.

Many simple robot designs will (attempt to) drive straight by using the technique of turning motors on either side at the same speed in opposite directions. To program motors to do this, choose motor directions which add up to 255. For example, if a CW motor is set at 190, the CCW motor should be 255 – 190 = 65.

## Multiple Programs

If you want to switch from one program to another without downloading each time, multiple programs can be embedded into a larger program and selected using jumpers (the small orange clips) placed into the digital inputs.

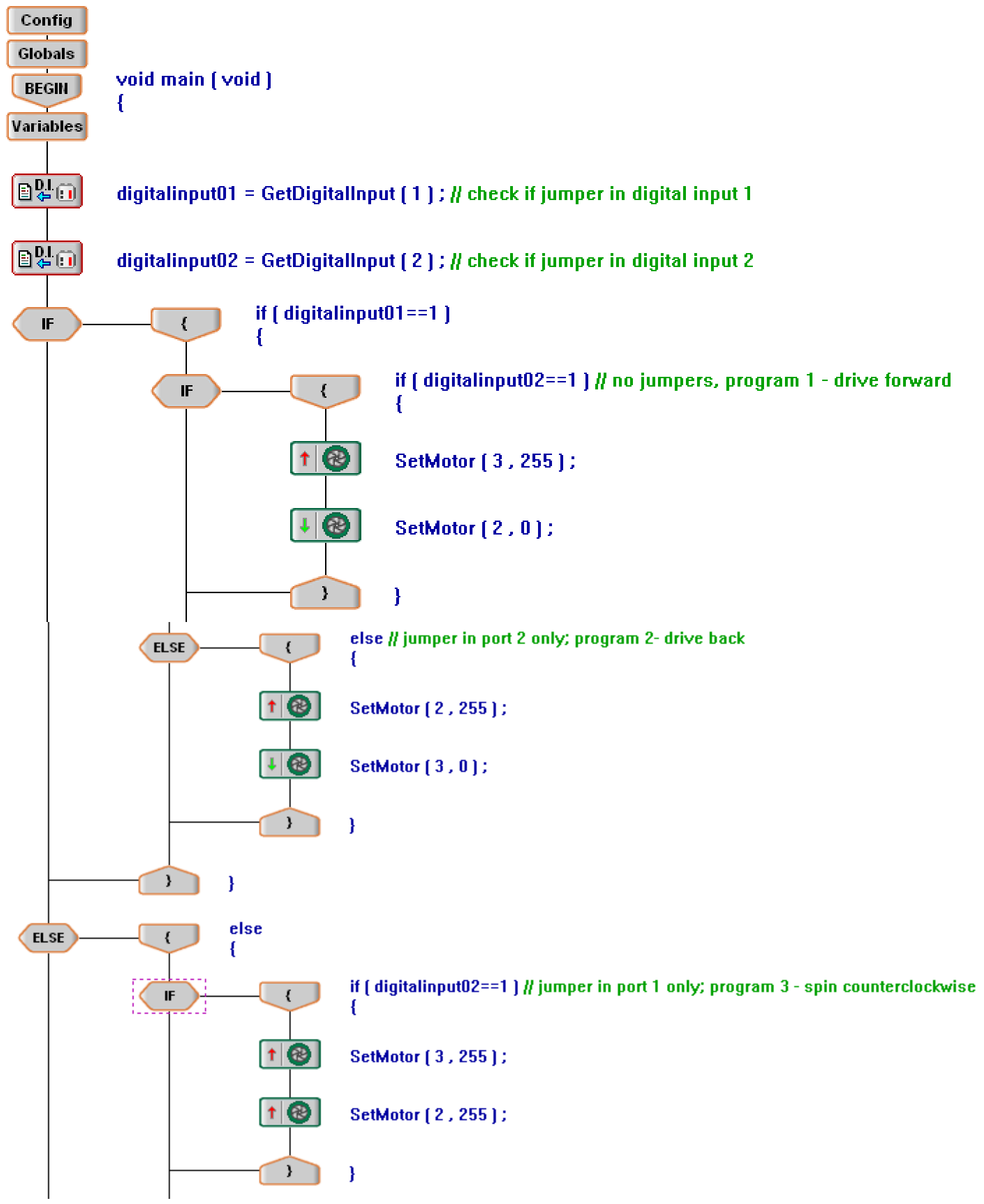
You can test the effect of jumpers on the digital inputs by using the Online Window (Inventor's Guide, p. 8.21). The window shows that ports 5 - 10 are configured as digital inputs. On the controller, there is a strip of slots labeled ANALOG/DIGITAL, number from 1 – 16, TX, and RX. Try inserting the jumper clip into the various ports, and note that for digital inputs (ports 5 - 10), the default configuration is "1"; placing a jumper in a digital input slot changes its value to "0."

One ergonomic modification is suggested here. I prefer to use ports 1 – 4 for jumpers, because they are on the very end of the port strip – ports in the middle are sometimes hard to see, especially if your controller is boxed in by other parts. If you are not planning to use analog inputs (or at least not the first few), you can reconfigure ports 1-4 as digital inputs. To do this, when you first open your program, double click on the gray "Config" button at the beginning of the program. You should see the layout of all the microcontroller ports. Right click on ports 1 – 4 to toggle between analog input and digital input.

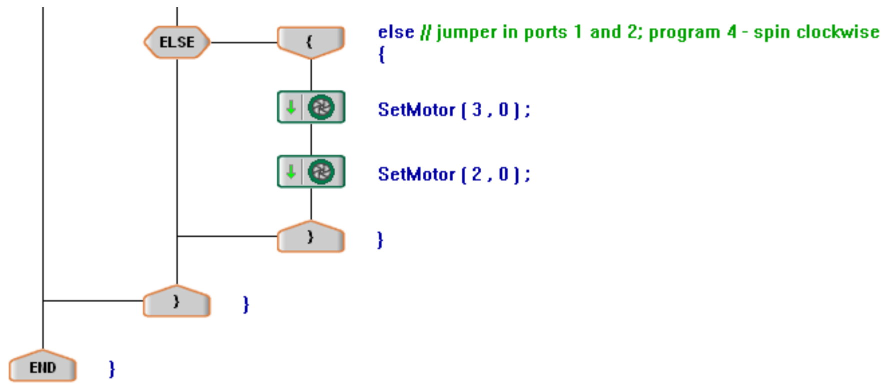
By reading the digital inputs at the beginning of the program, you can select up to $2^n$ different programs, where n is the number of ports used (think binary). For example, using 3 ports gives $2^3 = 8$ options in the following configuration:

| Program | Port 1 | Port 2 | Port 3 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |

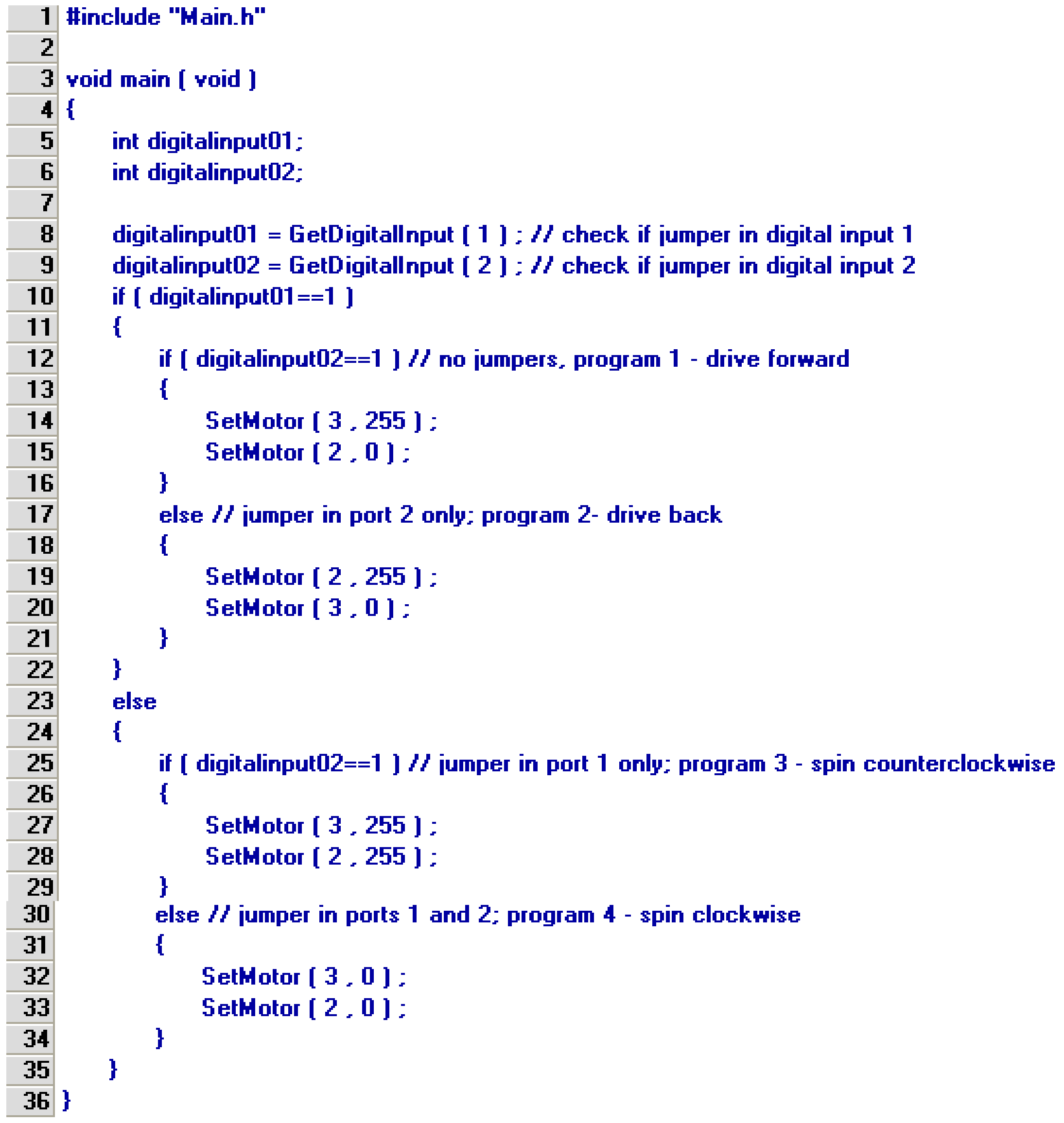
Below is a program that uses 2 digital inputs (inputs 1 and 2 are reconfigured as digital) for a total of 4 subprograms. To activate each program separately, insert the jumper(s) which will select the desired program before turning on the robot. The selected program will continue executing until the robot is turned off. You cannot switch from one program to another by switching the clips while the robot is running, unless you write your program in such a way that it continuously polls for the digital inputs in an infinite loop.



```
void main ( void )
{
digitalinput01 = GetDigitalInput ( 1 ) ; // check if jumper in digital input 1
digitalinput02 = GetDigitalInput ( 2 ) ; // check if jumper in digital input 2
if ( digitalinput01==1 )
{
    if ( digitalinput02==1 ) // no jumpers, program 1 - drive forward
    {
        SetMotor ( 3 , 255 ) ;
        SetMotor ( 2 , 0 ) ;
    }
    else // jumper in port 2 only; program 2- drive back
    {
        SetMotor ( 2 , 255 ) ;
        SetMotor ( 3 , 0 ) ;
    }
}
else
{
    if ( digitalinput02==1 ) // jumper in port 1 only; program 3 - spin counterclockwise
    {
        SetMotor ( 3 , 255 ) ;
        SetMotor ( 2 , 255 ) ;
    }
```



```
else // jumper in ports 1 and 2; program 4 - spin clockwise
{
SetMotor ( 3 , 0 ) ;
SetMotor ( 2 , 0 ) ;
}
}
}
```



```
#include "Main.h"

void main ( void )
{
      int digitalinput01;
      int digitalinput02;

      digitalinput01 = GetDigitalInput ( 1 ) ; // check if jumper in digital input 1
      digitalinput02 = GetDigitalInput ( 2 ) ; // check if jumper in digital input 2
      if ( digitalinput01==1 )
      {
            if ( digitalinput02==1 ) // no jumpers, program 1 - drive forward
            {
                  SetMotor ( 3 , 255 ) ;
                  SetMotor ( 2 , 0 ) ;
            }
            else // jumper in port 2 only; program 2- drive back
            {
                  SetMotor ( 2 , 255 ) ;
                  SetMotor ( 3 , 0 ) ;
            }
      }
      else
      {
            if ( digitalinput02==1 ) // jumper in port 1 only; program 3 - spin counterclockwise
            {
                  SetMotor ( 3 , 255 ) ;
                  SetMotor ( 2 , 255 ) ;
            }
            else // jumper in ports 1 and 2; program 4 - spin clockwise
            {
                  SetMotor ( 3 , 0 ) ;
                  SetMotor ( 2 , 0 ) ;
            }
      }
}
```

## Sensors

A robot can be programmed to go from one point to another simply by instructing the motors to turn on and off at the appropriate times, causing it to turn or go straight. Doing this is akin to attempting to drive to the store, blindfolded and with earplugs, hoping to stay on course by knowing the appropriate turns and distances. In real life, a driver is constantly monitoring the environment, listening for the sound of oncoming traffic, looking for obstacles, and braking when he feels the tires hit a bump. Adding sensors allows your robot to monitor and react to the environment in this way.

The programming software comes with test programs which demonstrate how to get sensor readings and show you the readings of the sensors onscreen. To access the programs, click File > Open Project > Test Code. From there, select the sensor you want to test.

### Bumper Sensor & Limit Sensor

The use of the bumper sensor to control the actions of the robot is described in detail in the tutorial that comes with the programming kit ("program three: using motors and sensors together," and "program four: reversing and turning," beginning from p. 8.38). The limit sensor is also digital (its only readings are 1 and 0, where 0 = pressed, 1 = released), which is why it is designated as a switch.

One of the disadvantages of using the bumper or limit switch alone to detect whether a robot has hit an obstacle is that the sensor itself must contact the obstacle. If another part, like a corner, of the robot bumps into something, the robot will continue to move forward. This problem can be rectified by "fronting" your robot with a wider bumper, which activates the switch when any part of the attachment comes into contact with an obstacle.

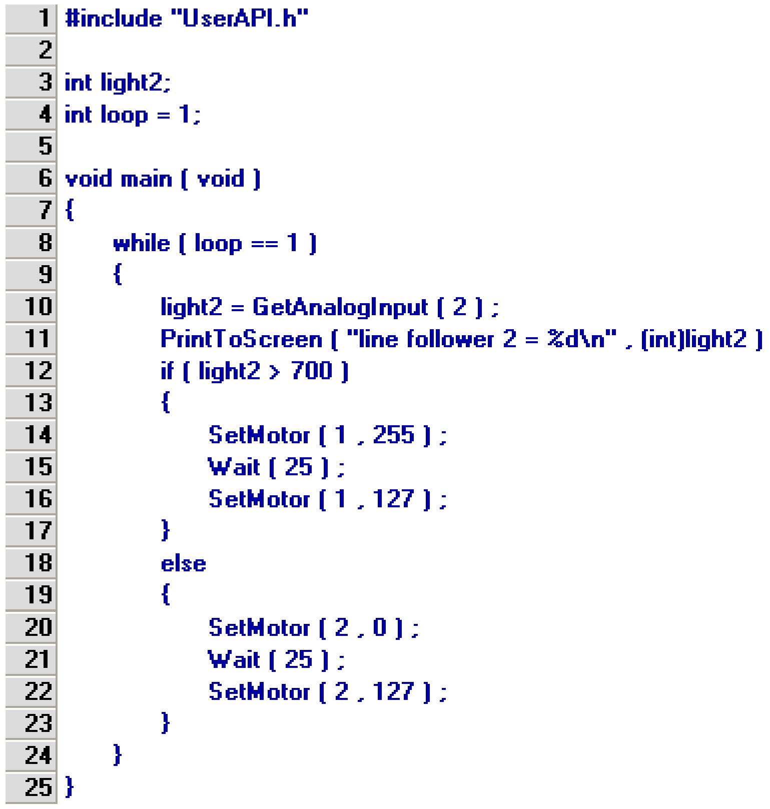
### Line Follower

The line tracker kit contains 3 sensors, but you don't need to use all three of them at once to follow a line. Suppose you want to follow a black oval ring on a white mat. You can use the LINEFOLLOWERTEST program in the Test Code folder to obtain sensor readings when the sensor rests on the black and on the white. These readings will fluctuate, depending on how close the sensor is to the mat. However, it should be clear which ranges of values indicate "dark" and "light," and a suitable threshold set to distinguish one from the other.

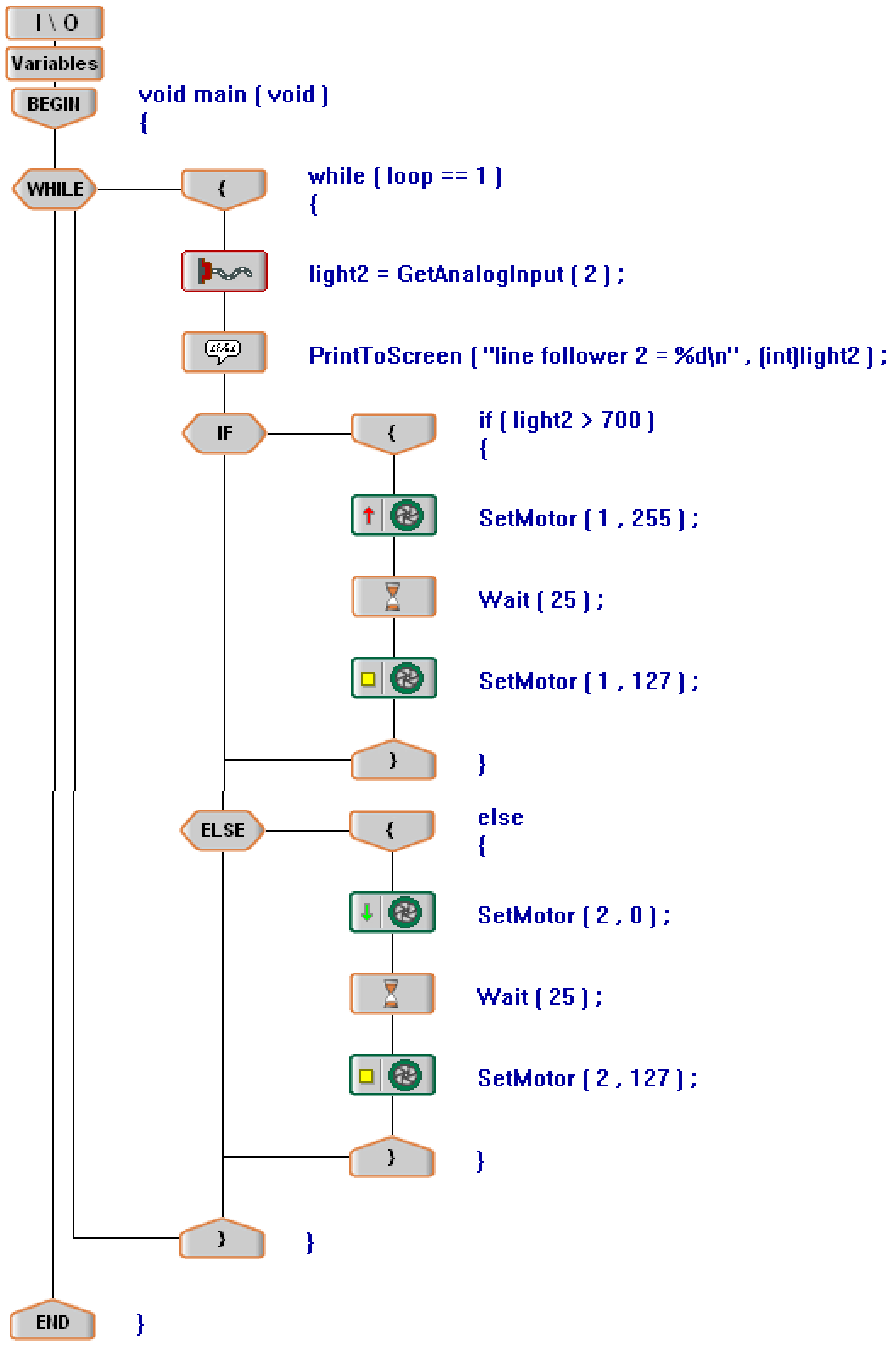
A one-sensor line follower actually follows the edge between black and white, rather than the line itself. It begins on one side of the line, say the right, and "looks for" the line on the left. When sensor readings cross the threshold from light to dark, the robot turns away from the line briefly, then turns toward it again, until it finds it. It continues to follow the edge between white and black, as long as the black is on the left and the white is on the right. The robot can also be programmed to follow the line starting from the left side.

Sometimes a line follower will spin in circles because it is looking for a line, say on the left, and the line isn't there. This indicates that the sensor has failed to detect the line; most likely the robot has driven over it without detecting it. This happens when the robot has driven too quickly or the sensor has "looked for" the line too infrequently. The problem can usually be rectified by slowing down the robot or decreasing the interval between sensor readings. Because of the overshoot dilemma, wide lines are easier to follow than narrow ones, and gentle turns are easier to follow than sharp corners.

The program below uses one sensor to follow a line on its right side. It uses the Slowerbot (see Locomotion section, 4-wheel drive) rather than the Squarebot to reduce the overshoot. A threshold value of 700 was chosen, since "light" readings ranged between 30 and 650, and "dark" readings were in the 800+ range.



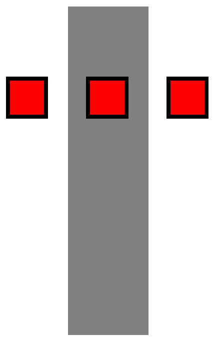
```
#include "UserAPI.h"

int light2;
int loop = 1;

void main ( void )
{
    while ( loop == 1 )
    {
        light2 = GetAnalogInput ( 2 ) ;
        PrintToScreen ( "line follower 2 = %d\n" , (int)light2 )
        if ( light2 > 700 )
        {
            SetMotor ( 1 , 255 ) ;
            Wait ( 25 ) ;
            SetMotor ( 1 , 127 ) ;
        }
        else
        {
            SetMotor ( 2 , 0 ) ;
            Wait ( 25 ) ;
            SetMotor ( 2 , 127 ) ;
        }
    }
}
```



```
void main ( void )
{
      while ( loop == 1 )
      {
            light2 = GetAnalogInput ( 2 ) ;
            PrintToScreen ( "line follower 2 = %d\n" , (int)light2 ) ;
            if ( light2 > 700 )
            {
                  SetMotor ( 1 , 255 ) ;
                  Wait ( 25 ) ;
                  SetMotor ( 1 , 127 ) ;
            }
            else
            {
                  SetMotor ( 2 , 0 ) ;
                  Wait ( 25 ) ;
                  SetMotor ( 2 , 127 ) ;
            }
      }
}
```

It should be noted that greater speed and precision can be obtained by using two or three sensors for line following. In a two-sensor configuration, the sensors straddle the lines. The left sensor "looks for" the line on its right and turns away to the left when it finds it; then, the right sensor "looks for" the line on its left and turns away when it finds it. When neither sensor "sees" black, the robot drives straight by default. Wider spacing between sensors allows the line follower to go faster, since it can continue in a straight direction until it hits the other sensor. Narrower spacing results in a slower robot, but one which follows the line with less side-to-side variation.

It is also possible to use three sensors[7], an arrangement commonly found commercially. The sensors are spaced so that at least one sensor will detect the line at any given time. If the left sensor sees black, the robot turns left until the center sensor sees black, and if the right sensor sees black, it turns right until the center sensor sees black. When the center sensor sees black, the robot goes straight by default.



Optical Shaft Encoder (rotation sensor)

The optical shaft encoder records the number of rotations that the axle turns in the shaft, where a reading of 90 = one complete rotation. This, in turn, can be used to calculate the distance that a robot has traveled by multiplying by the circumference of the wheel.

$$\text{distance traveled} = \frac{\text{sensor\_value}}{90} \times 2 \cdot \pi \cdot r$$

One unfortunate aspect of the shaft encoder is that it does not use positive and negative numbers to record clockwise vs. counterclockwise rotation. Thus, a reading of 90 might indicate a full CW rotation, a full CCW rotation, or a half-rotation CW followed by a half-rotation CCW. Thus, in your programming, it is very important to store the readings in a variable every time the direction of motion changes and reset the encoder when necessary.

Ultrasonic Sensor

The ultrasonic sensor detects distance by sending out a signal and recording the time it takes for the reflected signal to return. The number returned indicates the number of centimeters the robot is from a barrier. Moving robots produce significant fluctuations, as the distance changes from the time the signal is sent until the time it is received. An artifact of 99 (centimeters) periodically pops into the readings, especially if the robot is moving or changing direction, as the receiver cannot lock onto the signal.

Where necessary, programming can be used to filter out these readings. The documentation that comes with the ultrasonic sensor also describes how other factors, such as temperature and altitude, can affect the speed of sound. These will play a role in the readings obtained.

## Light Sensor

The light sensor differs from the line tracker, in that the light sensor is designed to read the intensity of visible light, while the line tracking sensors both emit and receive infrared light. Traditionally, the most common application of light sensors has been line-following, but a light sensor can also be used to detect light in a dark room, as well as distinguish between certain colors (which are read as darker or lighter shades of gray).

## Transmitter Considerations

Crystal Upgrade

Unfortunately, most Vex™ Starter Kits come with the same frequency module & crystal (#61), which means that signals from 2 remote-driven robots will interfere with each other, causing jerky, erratic motion. In fact, if you wish to use 2 transmitters to control different motors on the same robot, you will encounter the same problem. One solution is to purchase the Crystal Upgrade kit, which has sets of crystals and frequency modules tuned to different frequencies than the commonly used ones. Once replaced, this should prevent your robot from receiving the transmissions of other teams (unless by Murphy's Law, they have purchased crystals of the same frequency as you). This is a possibility, as there are only 2 Crystal Upgrade kits with a total of 8 different frequencies available at the time of this writing.

Tethering

One way to completely ensure that your transmitter will not interfere with another is to bypass the RF receiver(s) by transmitting signals to the robot via a tether. Using a phone cord (the curly cord, not the straight one going to the wall), connect the tether port on the back of the transmitter directly to one of the Rx ports on the controller. Since there are 2 Rx ports, you can connect 2 transmitters in this way. The downside is that you have to follow your robot around with the transmitter(s) and watch for tangling of cords. This is adequate for practice, but can't be used in most competitions. However, if you plan to attend a competition, it's advisable to bring a tether cord for use in the practice area, so that your robot's signals don't interfere with others and vice versa.

Programming the Transmitter (rather than the microcontroller)

Your transmitter can be programmed manually (without a computer) so that the joysticks and buttons operate differently from the default, as described in the *Inventor's Guide*, Appendix E – Control Configurations. However, reprogramming the transmitter should be done with caution, especially if you own more than one transmitter. Realize that switching from one transmitter to another (differently programmed) will cause your robot to operate differently when you activate the same controls; for example the robot might move forward when you push both joysticks forward on one transmitter, but spin when you push both joysticks forward using a different transmitter. Rather than having to learn new controls for every transmitter, you can save the driver some confusion by carefully labeling the transmitters or reprogramming every transmitter in the same way.

Alternatively, if you want the controls to operate in a certain way, you can modify the configuration through the software that you download. This way, all transmitters can be left in "default" mode and will produce identical results, preventing unwanted surprises.

## Simple Challenges

You can experiment with different robot designs by tackling some simple challenges which require little or no field construction. These are patterned after the Vex™ Pentathlon events of the Robofest competition (www.robofest.net). While they are optimal for two or more competing robots, one-robot challenges are possible.

### Tug of War

Attach a vertical partially threaded beam (for a "hitching post") to the chassis of each robot. Loop a rope around each hitching post, and see which robot drags the other across a line. For one robot, use a bungee cord, and see if you can increase the amount of stretch by modifying your robot. Alternatively, drag a weighted platform on the ground, and see if you can modify your robot to drag increasing amounts of weight

### Speed Race

Line up the robots at the starting block and let 'em rip. For one robot, time yourself for personal best. For an added challenge, use black electrical tape for the starting and finish lines, and require robots to start and stop at the appointed markers.

### Obstacle Course

Set up a maze or obstacle course using 2-liter bottles or soda cans. Maximum speed is desirable, with penalties for obstacles touched, moved, or knocked over.

### Race Track

Make an oval race track (butcher paper with black electrical tape works), and design the fastest line follower possible to make its way around the track.

### Skee Ball

Lob a maximum number of ping-pong balls into a desired target (like a waste basket) in a fixed amount of time (or a set number of trials). For a greater challenge, mount rectangular wastebaskets onto the rungs of a ladder, and assign points for each level.

## Further Exploration

Innovation First, Inc. (IFI)

IFI's website, vexlabs.com has photos and video clips of several interesting designs that demonstrate the extent of how much you can do with the Vex™ system. Check out the video clips at vexlabs.com/vex-robot-photos.shtml, and follow the links >Vex robotics – Index of Galleries > Robots > Ribbon Cutter Robot (additional robots can be found here as well). IFI is currently the distributor of Vex™ components, so they have a vested interest in providing ongoing support.

Chief Delphi Forum

Follow the thread chiefdelphi.com/forums/forumdisplay.php?f=146 to the Vex Challenge subforum on Chief Delphi. (The Vex forum also has good information, though it's newer and less developed than the Vex Challenge subforum). Posted here are a multitude of designs, building ideas, and answers to questions. Use the search function before posting a question – many of the good questions have already been asked. Better yet, read through all the posts on the forum before posting. To do this, you will need to display all threads from the beginning (not just the past month). Scroll to the bottom of the page, and under "Display Options" select "From the" > Beginning.

Vex Forum

The Official Community Site is located at vexforum.com. You can post a question, but questions are answered only by "the experts," not the entire forum community.

Robot Magazine

Every now and then, Robot Magazine (botmag.com), will have articles related to Vex™. Here's one:
botmag.com/articles/mythbusters_test_the_vex_robotics_design_system_1.shtml

*FIRST* (For Inspiration and Recognition in Science and Technology)

Information on the *FIRST* Vex™ Challenge competition can be found here: www.usfirst.org/vex/.

## References

[1]Innovation First, Inc. (IFI) vexlabs.com

[2]Chief Delphi forum (Vex subforum) chiefdelphi.com/forums/forumdisplay.php?f=146. Try the search function before posting a question!

[3]FIRSTwiki: www.firstwiki.org/index.php. If the domain name has expired, try http://wiki.chiefdelphi.com/index.php/Main_Page. FIRSTwiki has some good overviews of engineering concepts, but the coverage is spotty. Also, the information is geared more toward FRC (FIRST Robotics Competition) robots, rather than Vex™, though many of the concepts are transferable.

[4]Pilvines, John, and Team Unlimited (FVC 2013). FLL to FVC – a Case Study, Power Point Presentation, *FIRST* Robotics Conference Workshop, April 2007. You can download this presentation at: http://eaglevex.syraweb.org.

[5]Needel, Greg. Building Competitive Manipulators: The Mechanics and Strategy, Power Point Presentation, *FIRST* Robotics Conference Workshop, April 2006. Although this presentation was given by Greg Needel in 2006, he credits Andy Baker for much of the material, who credits Chris Husmann. You can download this presentation at: www.usfirst.org/robotics/2006/Workshops/wrkpresentations.htm.

[6]LinkageDesigner. www.linkagedesigner.com/res/fourbar.gif

[7]Dubel, William. Reliable Line Tracking. www.dubel.org

[8]Johnson, Joe. Useful Vex Tools, posted at: chiefdelphi.com/forums/showthread.php?t=37031&highlight=useful+tools.

## Appendix A: Cutting Suggestions for Commonly Used Pieces

1. 12" Square Bars (4-6)
   - Cut one 12" square bar into 4" lengths (3 pieces)
   - Cut three 12" square bars into one 4" length and one 8" length (2 pieces)
   - (Optional) Cut one or two 12" bars into 6" lengths (2 or 4 pieces)

   It takes 6 axles for a 4-wheel drive (4 for wheels, 2 for motors) like the Squarebot, and sometimes, the 3" axles are not long enough for optimum stability. The 8" and 6" bars are useful for manipulators.

2. 15" Angle Bars (8-12)
   - Cut 4 bars into one 10" and one 5" piece at the notch
   - Cut 4 bars into one 12.5" and one 2.5" piece at the notch
   - Keep 4 bars of 15" uncut

   It takes 4 rails of equal length to make a chassis which fully supports the gear train (2 rails on each side). It's nice to have the bars pre-cut for potential chasses of 4 different lengths (8" using the chassis bumper, and 10", 12.5" or 15" using angle bars). Since the Starter Kit only comes with 4 angle bars, 8 more will need to be purchased separately or as part of an upgrade kit. The leftover pieces (2.5" and 5") are very useful for manipulators and doodads – don't throw them away!

3. Long bars and plates
   It is recommended that you don't pre-cut the thin long bars and plates. These are easy to cut on the spot using tin snips, and they are less conducive to having frequently used, standardized lengths.

## Appendix B: Some Useful Tools (as recommended by Joe Johnson)[8]

These tools are available at McMaster.com at the prices and part numbers shown below (subject to change).

### Nut Driver for 8-32 Nuts:

8358A26
Hollow-Shaft Inch Nutdriver Standard, 11/32" Size, 3" Shaft Depth, Green
In stock at $4.52 Each

This tool makes it easier to tighten a nut in close quarters. You can often get this nut driver in places that the wrench just won't go.

### Hex Driver for 8-32 Screws:

5497A68
Comfort Grip Ball Driver Long Length, 5/64" Hex, 4-21/64" Blade Length
In stock at $2.33 Each

### Hex Driver for 6-32 Motor Screws and Shaft Collar Set Screws:

5497A69
Comfort Grip Ball Driver Long Length, 3/32" Hex, 4-51/64" Blade Length
In stock at $2.39 Each

These two tools save TONS of time when building, repairing, etc. In the pits in Atlanta, you are almost certainly going to be taking your robots apart between rounds for this or that reason. Saving time means you will reduce the general stress level. HIGHLY RECOMMENDED.

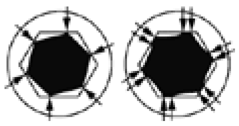
### Hi Torque Hex L-keys:

71285A151
Hi-Torque L-Key 5/64" Hex, 4" Length
In stock at $0.24 Each

71285A152
Hi-Torque L-Key 3/32" Hex, 4-13/32" Length
In stock at $0.27 Each

These L-keys have a profile that make a significant difference when you are really tightening a screw. They do a much better job of keeping the hex from stripping. These things are cheap. Buy 10 of each of them. Use them until they start to show wear and THROW THEM OUT – you'll be glad you did. It is better to use fresh tools and not strip a screw. At about a quarter each, it is money well spent.

Regular key on left - Hi-torque key on right



## If you are going to replace motor screws…
96452A144
Torx Button Head Socket Cap Screw Alloy Steel, 6-32 Thread, 1/4" Length
In stock at $12.43 per Pack

96452A148
Torx Button Head Socket Cap Screw Alloy Steel, 6-32 Thread, 1/2" Length
In stock at $13.16 per Pack

If I had more time, I would find these screws in stainless steel or with a zinc finish so that it would be easier to sort the motor screws from the 8-32 construction screws, but McMaster has only black oxide finish available.

## Torx Driver for Above Screws:
5756A14
Torx Driver Size T10, 3-1/4" Blade Length, 5-5/8" O'all Length
In stock at $3.20 Each

This tool will save time for building and repairs.

## Torx L-key for Above Screws:
6959A41
Torx L-Key Long Arm, Size T10, 3-3/8" Length
In stock at $0.95 Each

This tool is another one that I would buy several of.  Torx tools are nowhere near as likely to round out or strip, so I would perhaps only buy 5 of these where I might buy 10 of the similar hex tool.

## Tools for Torx Motor Screws:
37455A22
Comfort Grip Torx Ball-Point Driver T10 Torx, 2-23/32" Blade Length, 5-3/16" L O'all
In stock at $6.40 Each

This is a nice tool, more expensive, but the ball end feature makes it possible to tighten some screws at "off angles" that you just can't get to with a normal tool.

## <u>Appendix C: Recommendations on Additional Accessories</u>

Once you get started, it's easy to spend a lot of money on accessories, and the ones you select are largely dependent on the project you plan to undertake. If you are building a robot for competition, high on your list will be any accessories that are required or permitted for that competition. However, based on our own experience and that of a number of Vex™ users, the following accessories are recommended, arranged in groups, with most universally used items first.

<u>Tier 1 - The "Must Haves"</u> - These come in the FVC bundle pack.

- Power Pack - This will save money on batteries in the long run.
- Hardware and Metal Kit (or upgrade to the Tri-Pack Bundle below) - Replacement parts for the ones you cut.
- Programming Kit - Without programming, the Vex™ kit is basically just a remote-control vehicle kit. The programming kit is required for all sensors and allows you to create an autonomous (self-guided) robot. It also has an EasyC tutorial, which provides guided instruction in the use of your Vex™ kit.

<u>Tier 2 – High Demand "Extras"</u> – These are items we frequently wished we had more of. They are especially useful if you plan to focus on the mechanical, rather than programming, aspect of robotics.

- Gear Kit(s)
- Motor Kit(s) (for extra Motor Modules)
- Extra PWM extension cables (for connecting motors mounted far from the controller). These come in 4-packs, lengths of 6", 12", 18", or 24"
- Extra 7.2 v battery - Allows one to use while you're recharging

<u>Tier 3 – Other Desirable "Extras"</u> - There is much disagreement about which items should go into this category, but these items seem to be the most popular, after the Tier 1 & 2 items.

<u>Mechanical/Electrical</u>

- Wheel Kit – needed if you plan to use 4 medium or 4 large wheels
- Omni Wheels (large and small) – versatile & good for castors
- Tank Treads
- Chain & Sprocket Kit
- Advanced Gear Kit
- PWM Y-cables – For controlling 2 motors with the same input
- An extra starter kit - This is an economical way to go if you plan to buy extra wheels, motors, gears, metal and hardware, and remote.
- Tri-Pack Bundle - Contains Advanced Motion, Metal and Hardware kits with many parts for flexible building
- Chassis Kits - For larger robots

Sensors

- Line Tracking Sensor Kit – has 3 infrared emitters/sensors.
- Optical Shaft Encoder Kit – Contains 2 rotation sensors, useful for mapping distance
- Ultrasonic Range Finder – Commonly used for detecting proximity to (and avoiding) barriers.

Tier 4 – Specialized Items

- Crystal Upgrade - Necessary if you plan to use 2 or more remotes at the same time
- Light Sensor - Gives (visible) light intensity readings. For line following, the line tracking kit is recommended.
- Limit Switches – Two of these come in the starter kit
- Bumper Switches - One of these comes in the starter kit
- Pneumatics Kits - Very cool, but costly
- Servo Kit(s) – Limited to 120$^{o}$ of rotation, they're not practical for manipulators that are heavily geared down.
- Swerve Drive Kit