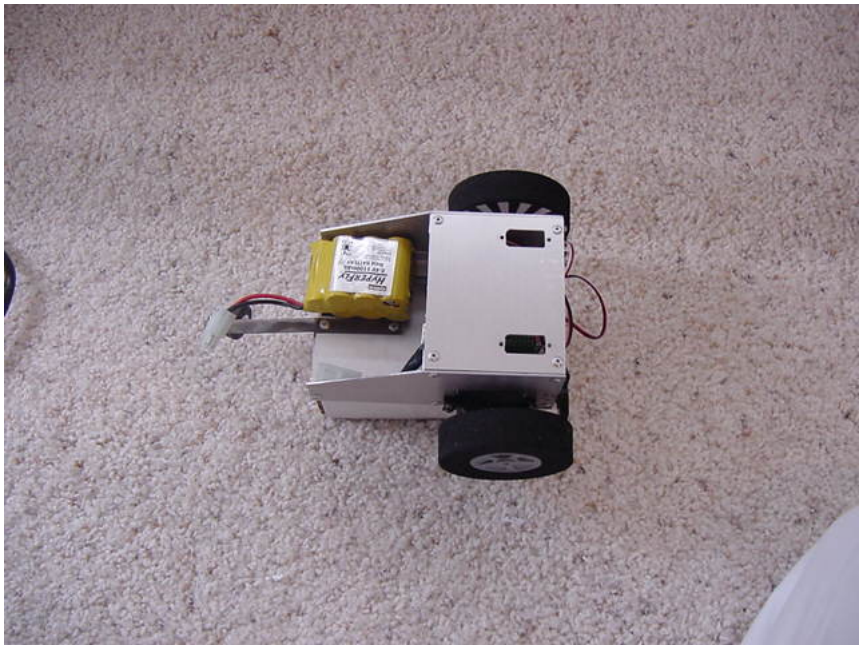


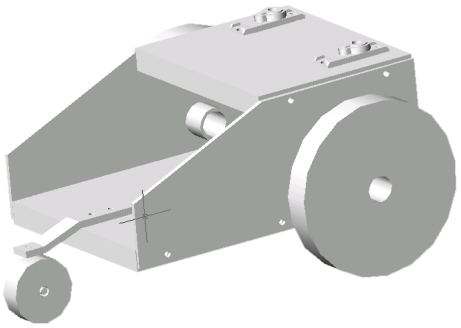
Sparky was crated for the June, 2 2001 robotics competition (See Pictures) in San Diego (SD RS events). The event was to go around 4 obstacles and return to the exact starting position. There was an advance round in which the robot has to circle the obstacles and also come back to the same starting point. I only participated in the first event. The robot did OK, it got second place (Well out of 2 robots that enter the competition. Mine and Alex Brown). To determine the distance to the starting point I was doing dead reckoning and sonar reading in which I tried to mach the sensor values obtain when the robot first started. But the timing for the dead reckoning was off and the robot kept on missing the target. Well, there is always next year. Now I use the robot as an EBOT for the Robotics Society of Southern California .

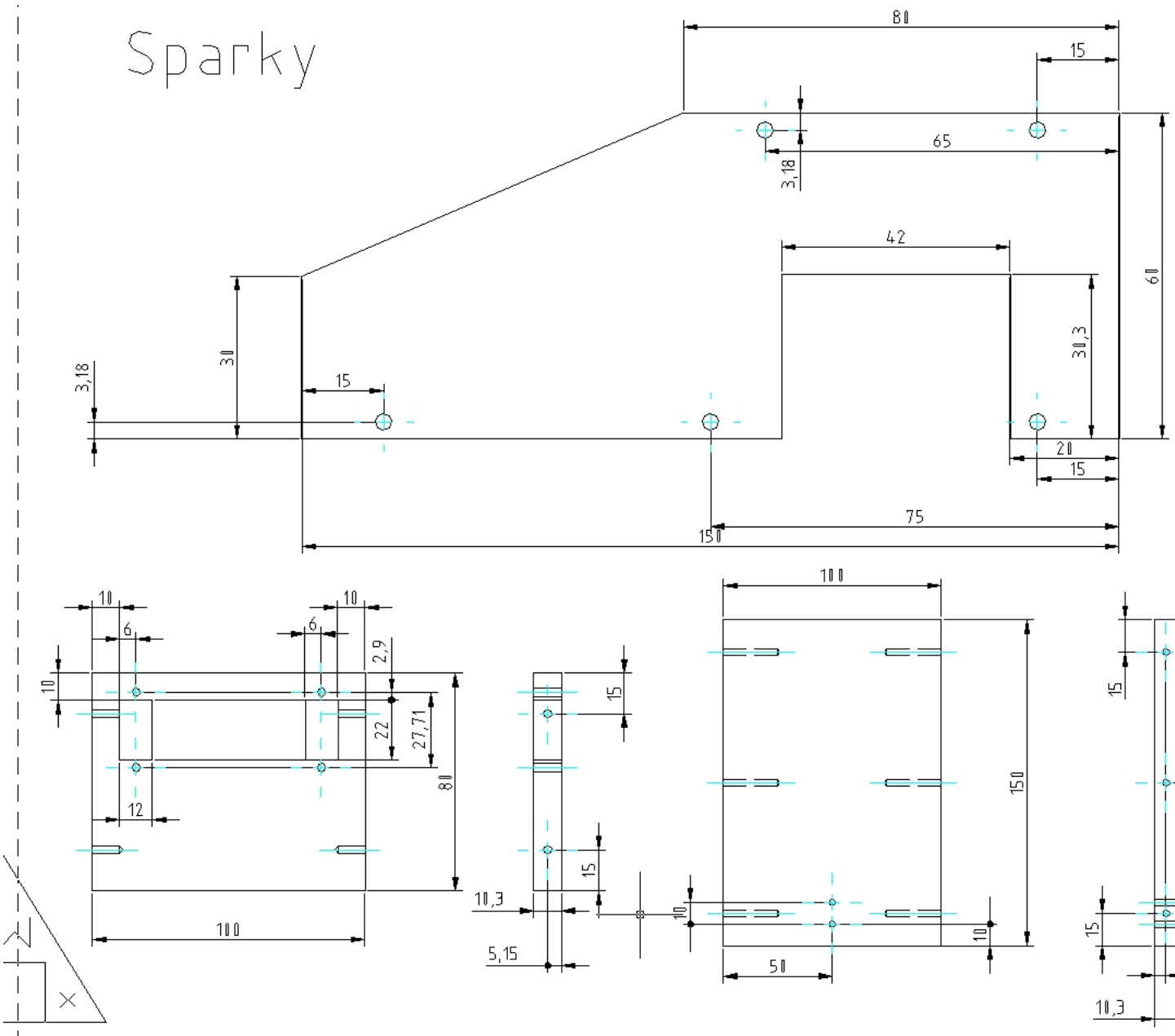
Here is a detail explanation of how I build the robot and the algorithms.

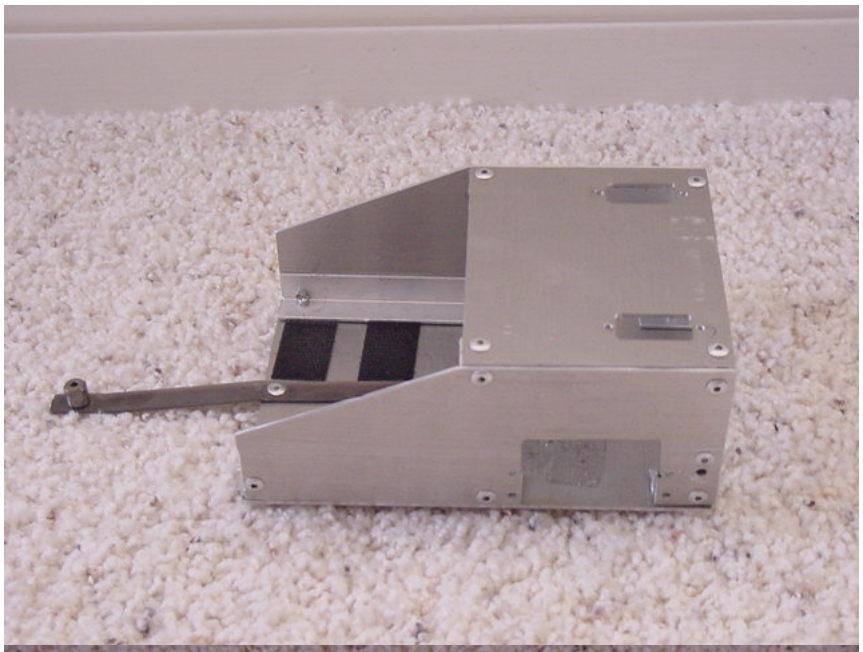
Sparky learning to follow a wall using a neural network

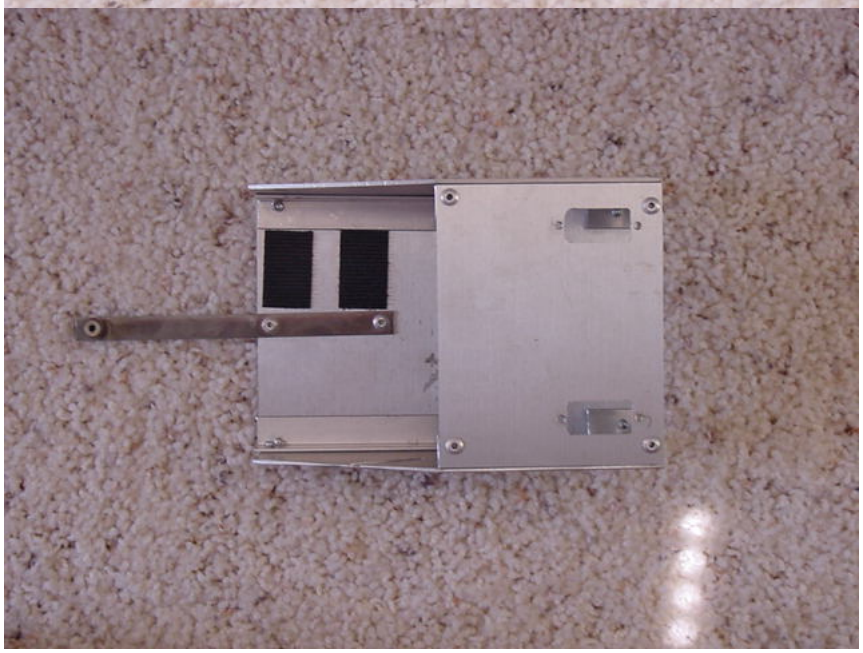
For more info see Introduction to neural network.

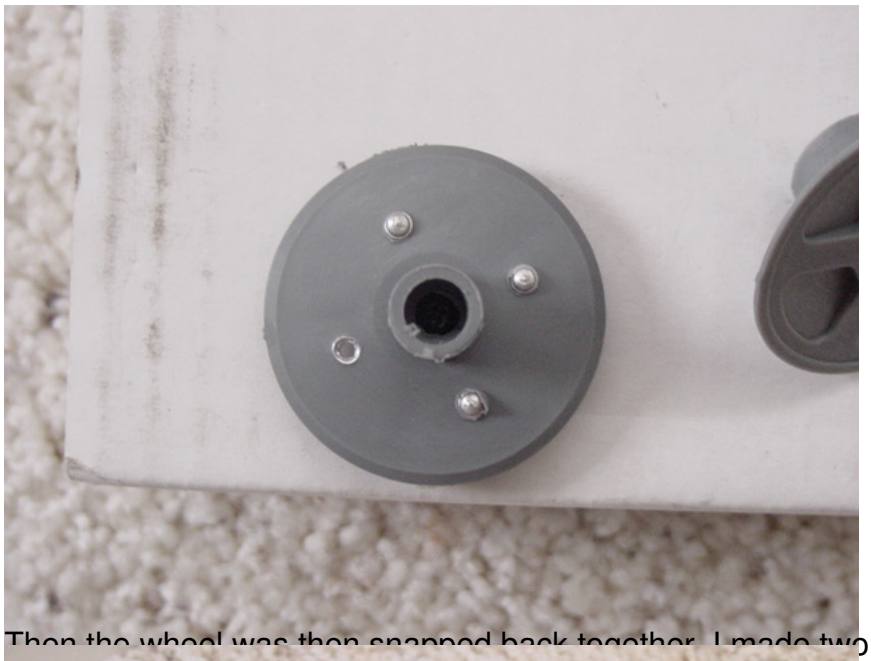
{youtube}1T7xi8jCgOE{/youtube}




Sparky
80
15
65
3,18
42
60
30
30,3
3,18
15
20
15
75
150
10
6
10
6
2,9
10
22
27,71
80
12
100
15
15
10,3
5,15
100
150
10
50
15
15
10,3
5

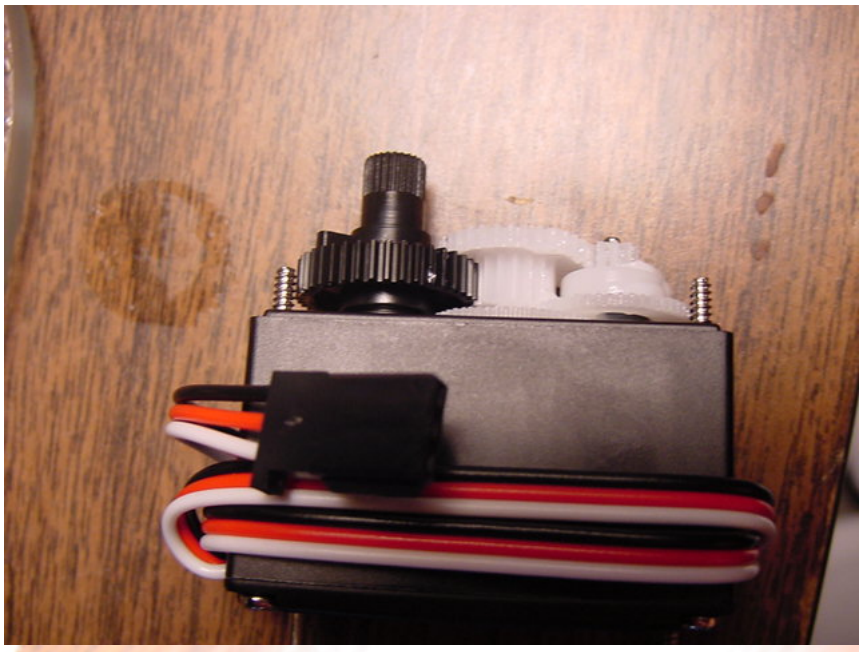




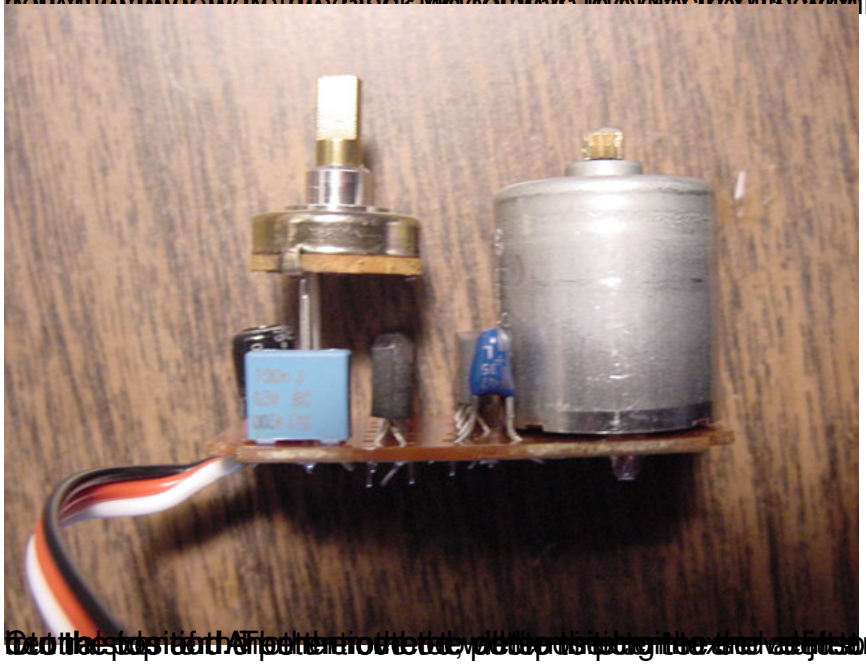


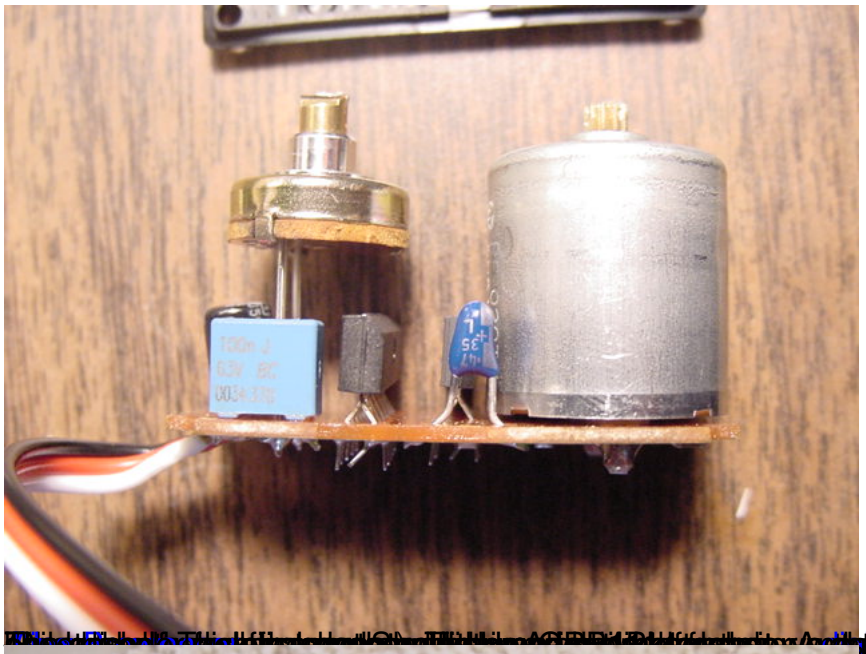
Then the wheel was then snapped back together. I made two of these wheels.

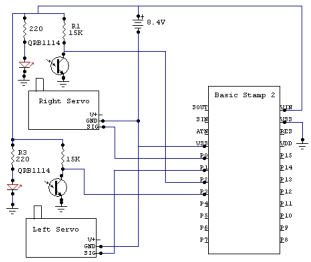
Open the top of the case and remove the black gear.



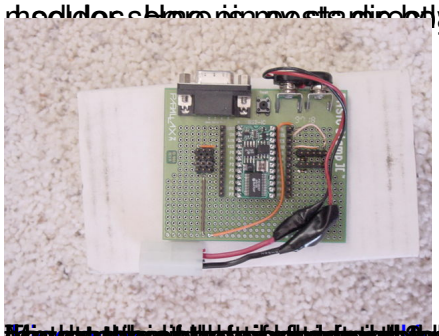
Cut the plastic tab in the black gear off.

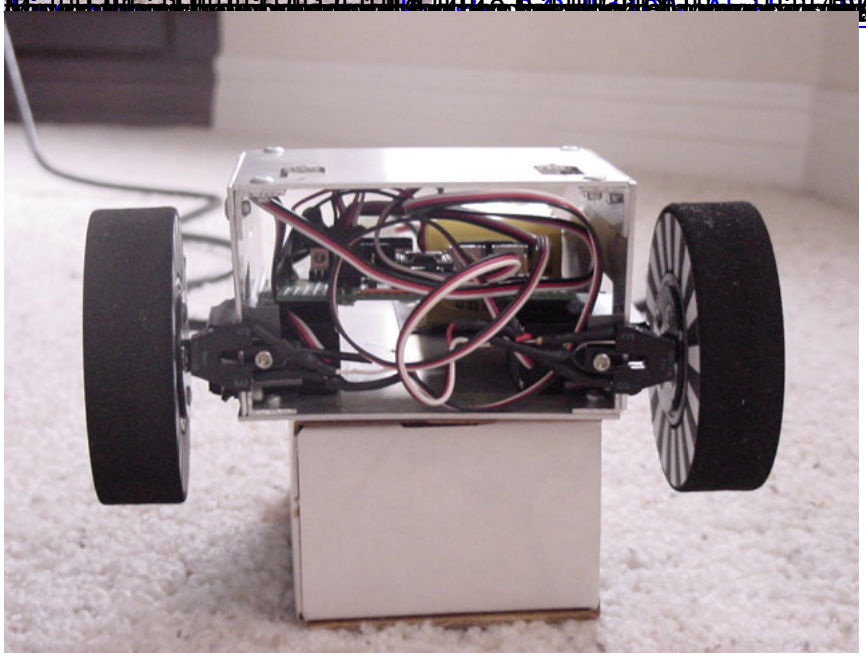


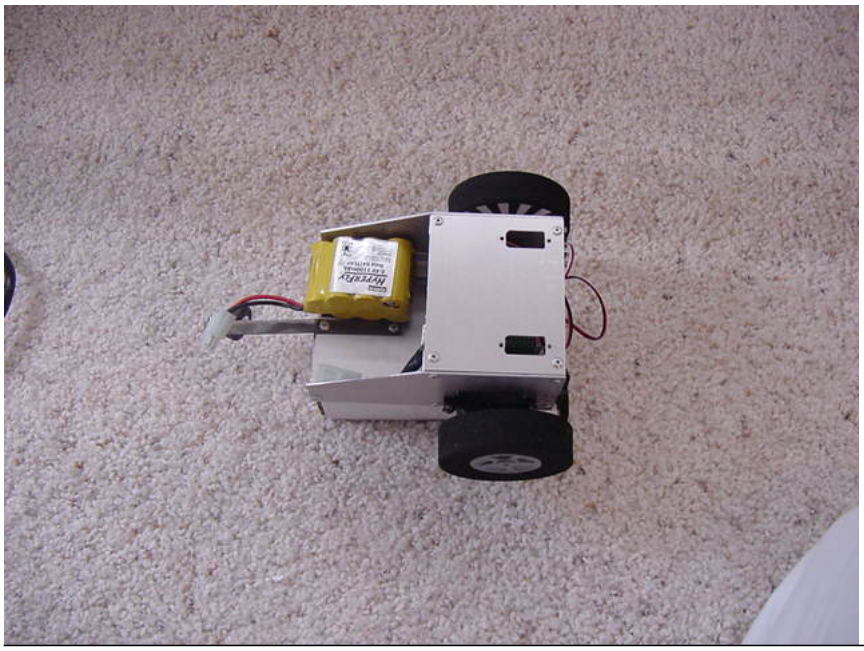




the stamp board where I plug in the servos or any other







Start Program

simple.bs2

This simple program will get the two motors running at predetermine speed. This program is also used to determine the right and left wheel stop values. You should place the robot on a box where the wheels can spin freely. Run the program and set the speed to 0. Keep modifying the values for Right_Stop and Left_Stop until both of the wheels are stopped. Remember these values because all the programs I use, use these values.

Reading Encoders

These simple programs are test programs for the encoders. And some more advance navigation.

reading_encoders.bs2

The first program will drive the wheels at some speed and will show the encoded speed on the debug window.
The wheels I am using came out to be 3⅛" or 3.125". And the encoders have 32 tics for the each wheel. So the distance traveled in or revolutions (32 tics per rev) is found based on the circumference of a circle.

C=2πR or C=πD.

C=π(3.125).

And to calculate the number of inches per tic.

IPT = C/tics = π(3.125)/32 = 0.3068 inches/tics

simple_nav.bs2

The next program will make the robot go ~11 ft, rotate 180 deg, and come back, while maintaining both wheels velocity the same. This will hopefully make it go straight because as the robot turns, one wheel has to go faster then the other.

Here are videos of the alg in action.

{youtube}msWM1ni3b-0{/youtube}{youtube}waIyc3lPHzU{/youtube}

A few problem with this algorithm in this particular robot.

1) As the wheels change speed the robot will try to correct the speed not the heading. So the the robot will end up going straight slightly off track.

2) As the robot makes the 180 deg turn to the right, it leaves the caster wheel pointed to the right. This will push the robot to the right until the equations kicks in. But because of the first problem the robot will be going to the right off the original course. I corrected this by not

completing the turn and making only 160 deg and the robot return pretty close to home.

The next method I will try to implement a pid base on position and not velocity.

June 2, 2001 Contest

jun2_nav4.0.bs2

The event was to go around 4 obstacles and return to the exact starting position. There was an advance round in which the robot has to circle the obstacles and also come back to the same starting point. I only participated in the first event. The robot did OK, it got second place (Well out of 2 robots that enter the competition. Mine and Alex Brown). To determine the distance to the starting point I was doing dead reckoning and sonar reading in which I tried to mach the sensor values obtain when the robot first started. But the timing for the dead reckoning was off and the robot kept on missing the target.

This is the program I used. It requires two ultrasonic sensors positioned at 90degs so one reads the values in from the front and the other reads the left side value. I decided to make the robot rotate to the left to provide some change, since I thought every body will make theirs rotate to the right (I don't know why it's that). The code is not pretty by all means.


NO