Now that you have a working robot, it's time to determine its navigation characteristics. The exercises below will demonstrate how your robot behaves when it turns and drives. This information will prepare you for a dead reckoning competition.

Driving and Steering Robots

The CheapBot robot base has two motors and wheels. For stability, a third point, the tail dragger, is located as the rear of the robot base where it behaves like a tripod. To make a CheapBot robot drive forward, both the left and right motors must rotate in the forward direction (that is from the robot's perspective). From the servo motor's perspective, one is turning forward and the other is turning backwards. However, since the motors are oriented opposite to each other, the wheels are turning in the same direction from the robot's perspective and the robot travels forward. If both motors reverse their spin, the robot still travels in a straight line, but this time backwards.

To turn the CheapBot robot, one wheel must turn in the opposite direction from the other. In other words, one wheel spins in the forward direction and the other spins in the backward direction. If you reverse the direction of spin of both motors, the robot still turns, but this time in the opposite direction (clockwise versus counter clockwise).

The H-Bridge takes two commands from the robot controller and converts it into a direction of rotation for the motor connected to it. The commands are how you set the two I/O pins connected to it. On the CheapBot-14, the two I/O pins connected to the first H-Bridge is 2 and 3. For the second H-Bridge, the I/O pins connected to it are 4 and 5 (on the CheapBot-18 the I/O pins are 0 and 1 for the first motor and 2 and 3 for the second motor). You send commands to the H-Bridge by setting the two I/O pins connected to it to either a HIGH (+5 volts) or LOW (0 volts). Since there are only two setting for each I/O pin (either HIGH or LOW) and there are two I/O pins, there are four combinations or commands you send to the H-Bridge. A truth table for the H-Bridges will look like this.

B.2 State	B.3 State	Rotation Direction
low	low	no rotation
low	high	clockwise
high	low	counter clockwise
high	high	brake

After trying to drive one wheel of your robot, its time to make your robot drive forward, drive backwards, turn left, turn right, and stop. Type the following commands and record which direction your robot drives.

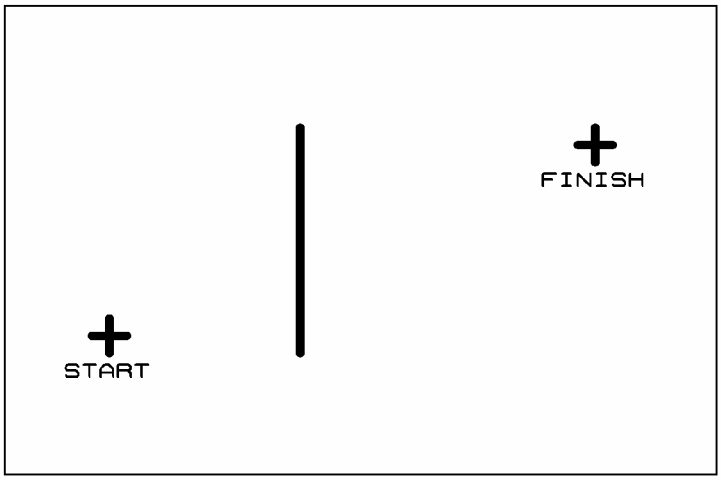
B.2 State	**B.3 State**	**B.4 State**	**B.5 State**	**Robot Direction**
low	low	low	low	
low	high	low	high	
low	high	high	low	
high	low	high	low	
high	low	low	high	

Dead Reckoning Navigation

The simplest way to get your robot to drive from one location to another is by dead reckoning. Dead reckoning means navigating your robot by specifying a direction and a pause length. The robot receives no feedback as to its position or heading. Therefore, when robots navigate by dead reckoning, they tend to drift out of the place you expect. Careful programming will help limit the drifting, however, there are too many unpredictable behaviors in even the simplest robot. Try out Project 1 to see this for yourself.

Project 1
Dead Reckoning by a Roboticist

With tape or a cone, mark a starting point on the floor. Then with a second piece of tape or cone, mark a destination about ten feet away. Between the starting and finishing points, place a barrier, such as tape, strip of wood, or cones, which you're not allowed to walk over. The exact placement is not important, just as long as you can safely walk between the start and finish points. Your test arena should look similar to this.



Have a friend with you, as you're going to pretend to be your robot and you'll need someone to look out for your safety and to keep track of how well you did. Now, walk from the Starting Point to the Finishing Point. As you do, keep track how many steps you took and when and how you made a turn. For example, your walking directions could look something like this,

Walk six steps forward
Turn 90 degrees to the left
Walk four steps forward
Turn 90 degrees to the right
Walk three steps forward

After you memorize your directions, walk back to the Starting Point. Orient yourself like you did the first time and then close your eyes. Following your directions, walk to the Finishing Point without opening your eyes. Your friend will make sure you don't walk into something.

After you get to your destination, open your eyes and see how close you are. Ask your friend if you crossed the tape barrier that you were not supposed to cross.

How close were you to the Finish Point?

Did you avoid walking across the tape Barrier?

This is the problem your robot faces when it dead reckons. It will drift out of line as it drives. Without a control loop that looks at the robot's current position and heading, there is no way to correct the errors that will build up. However, the Dead Reckoning competition asks you to understand the mechanical behavior of your robot the best you can. To have the best chance to dead reckon through a maze to your destination, you'll carry out the next two projects.

Project 2
Time versus Distance: How far will the robot go?

Your robot drives because your program calls a subroutine. The subroutines used in navigation set the four I/O pins connected to the H-Bridges to either +5 volts or ground (or called high and low). When the I/O pins are set, they trigger electronic switches inside the H-Bridges and those control the direction in which current flows through the motor attached to them. To keep the motors turning, you just need to leave the H-Bridges alone. One way to do this is to use a pause command (there are other ways and you will use one with the line follower). The pause command stops the robot controller from doing any other work for the length of time specified. So if you typed the following commands…

```
GOSUB FORWARDS
PAUSE 1000
GOSUB FREEZE
```

The robot will drive forward for one second until commanded to stop. Just how far your robot will drive depends on the charge in its motor batteries (the logic batteries don't affect how fast or far the robot will drive) and the size of the wheels (bigger wheels make the robot travel faster and therefore farther). To successfully compete in the dead reckoning competition you need to know how far your robot will drive with a given pause. So type the following commands and your FORWARDS subroutine in the PICAXE Editor.

```
PAUSE 1000
GOSUB FORWARDS
PAUSE 500
GOSUB FREEZE
END

FORWARDS:
        :
        :
        :
        :
RETURN
```

Note: You must type the four HIGH/LOW commands of your Forwards subroutine where the colons are located.

The first pause command is a delay to let you get your hands away from the robot before it starts up. So don't change the **PAUSE 1000**. The second pause is the delay that allows your robot to drive forward. In this project, you'll change the length of this delay and measure how far your robot drives.

- Make sure there is a space of at least five feet or more that the robot can drive down
- Mark the beginning of the range with a piece of tape
- Have a yard stick or tape measure handy
- Program your robot with the code above (with the motor power turned off)
- Turn off the logic power and unplug your robot
- Place the front of the robot at the tape mark
- Power up the motor power
- Power up the logic power
- After the robot stops, measure how far it traveled
- Enter the distance in the table below

PAUSE	DISTANCE
500	
1000	
1500	
2000	
2500	
3000	
3500	
4000	
4500	
5000	
5500	
6000	
6500	
7000	
7500	
8000	
8500	
9000	
9500	
10000	

- Change the 500 millisecond delay to 1000 and repeat the test
- Record the distance your robot traveled
- Repeat by adding 500 milliseconds to the pause length and measuring how far the robot drove

Once you're done, transfer the results you recorded into the graph at the end of this project. Keep this project with you, as you'll need this data for the dead reckoning competition

Project 3
The Perfect Right Angle

Turns for the dead reckoning competition only need to be right angles (right angles are 90 degree angles).

Right 90 degree turn needs a pause of:
Left 90 degree turn needs a pause of: