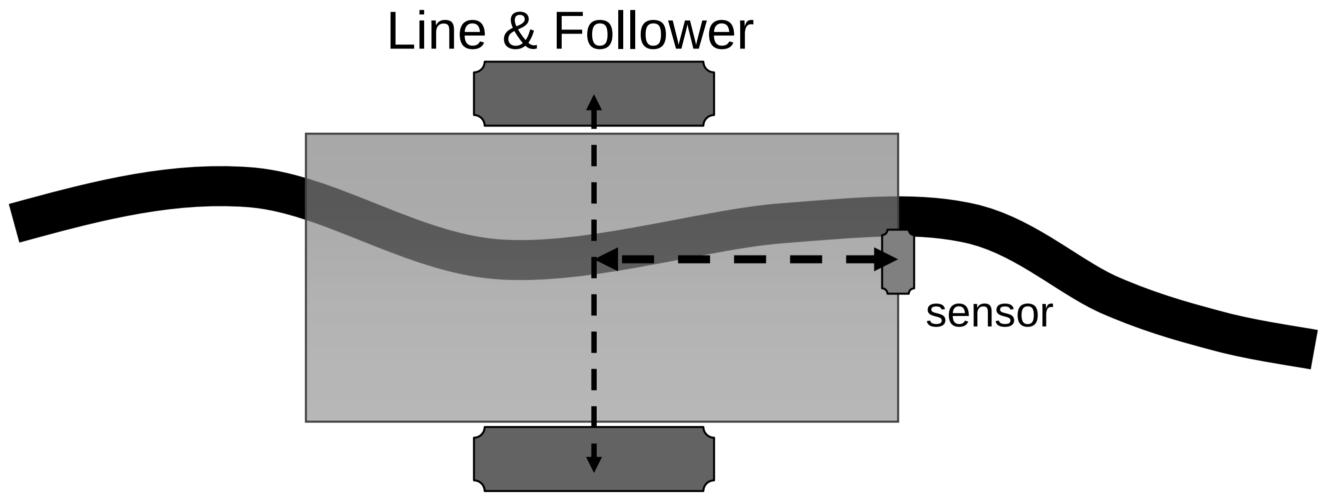
Line Following Task

Many introductions to robotics programming include an example of "Line Following" code using one or two sensors. This may seem appropriate since the sensing is simple and the code can be written with very few lines. However, this behavior is *not* simple and describing the task is not easy. [good programming must start with a well-defined objective.] If you search the web for examples you will find many from reports by 5th grade students using a Mindstorms kit, to descriptions of major open competitions and various winning strategies. An unconstrained requirement to "do rapid, accurate line following" is an invitation to a whole range of topics, most of which are beyond the concepts for beginners.

Let's try to explain the basic line-following behavior objective, assuming a single smoothly-curving black line (3/4" electrical tape) on a white background, and a single reflectance sensor on a two-wheel, drive-steered robot. The task implies a behavior where robot moves forward while guiding itself to follow the curve of the line.

A drawing showing the sensor, pointed down at the path and its location in relation to the drive wheels is important to understanding the task.
With a single sensor, the guiding signal is the difference between the black line and the white background. The objective would be to maintain a gray level indicating that the sensor is at the edge of the line, following either the right or left edge as the robot moves forward. Let us assume a configuration as shown from a top-down view:



Two dimensions are important to guiding the robot, the wheel-base, or axle, and the distance from the sensor to the axle. As the robot turns slightly, one wheel moves ahead of the other, and the sensor moves toward or away from the line in proportion to ratio of the dimensions D/A, where D is the distance from the axle,

and A is the length of the axle. (If the sensor is close to the axle, the motion will be smaller.*)
Now we can define the behavior in more detail: The robot must be controlled to move forward at a rate *and* to turn as directed by the reflectance sensor. The turn is the tricky part and requires a control loop to rapidly decide the magnitude and direction. The rate of turning is adjusted, at sample intervals, in response to the curvature of the line with a magnitude determined by the speed forward, the angle of the line change, and the D/A ratio (or Y/N decision based upon the sample). Finding the optimum control is the subject of a course in feedback control, but a simple, workable method can be found by trial and error: If you have too much turning the robot will be moving back-and-forth across the line faster than it is moving forward, and it may jump beyond the line edge (if the turning movement is too large for a sample interval). If you have too little turning the robot will follow slight line curves, but not keep up with a sharper turn, and thus will lose the line edge.

Various non-linear strategies can be tried to allow more drastic turns only when needed, such as: 1.pivoting or spinning if the line edge isn't found in one or two samples (measured by sample counts or clocks) 2. slowing the forward motion if the line edge hasn't been found quickly 3. Turning a fixed amount and moving forward a fixed maximum amount if the line edge hasn't been found quickly. 4. Etc. [This, along with the infinite range of D/A ratios and speeds, and the variables in line extremes, leads to many opportunities for creative designs and also many 'line Following' competitions.]

Another issue, which the developer will soon be faced with, is the context of the desired line-following behavior. In a Botball competition line-following is only the selected means to the end of arriving at a scoring object or a target location. How does the robot transition to and from line-following? To answer this question a method for creating the 'start' and 'stop' of the line-following behavior must be developed. Often an additional sensor is sampled to decide.

For further discussion and a sample code for test/calibration, check on
http://nasarobotproject.wordpress.com