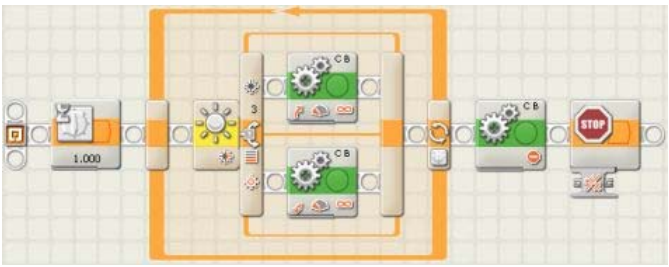
Line Follower NXT-G Program

A Simple Line Follower Program



The above program works fine as long as things stay "predictable" That is, as long as our course, or black line, always turns in the same direction: like a circle. Of course, you may find that your robot always wants to go in the same direction around the circle! The big problem occurs if your track, or black line, has "S" curves in it. There you may find your robot wants to reverse direction and retrace its path the way it came! A way around this problem is to use more advanced programming techniques.

A More Advanced Line Follower Program

A more complex line following program uses a "state machine" approach. In this case a "Mealy" state machine where the next output depends both on the current state of the machine, as well as the inputs. The current state of the machine is the length of arc, or circle moved in the opposite direction during the previous attempt to find the line. The input, of course, is the input from the light sensor, which can reset the state of the machine. The algorthm works as follows:

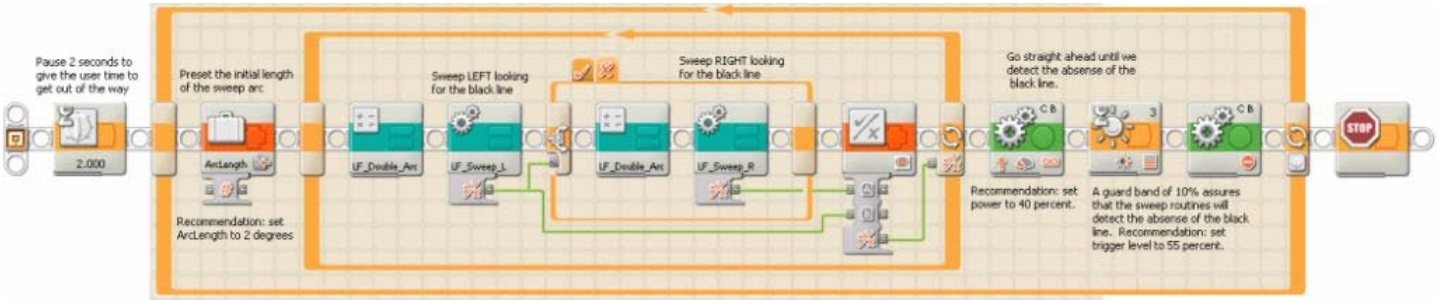
```
1. Reset the sweep variable to two.
2. Double the value stored in the sweep variable.
3. Unless the line is found, turn clockwise the number of degrees
   stored in the sweep variable.
4. If the line is found then goto step 9.
5. Double the value stored in the sweep variable.
6. Unless the line is found, turn counter-clockwise the number of
   degrees stored in the sweep variable.
7. If the line is found then goto step 9.
8. Goto step 2.
9. Drive straight forward while the line continues to be
   found.  If the line ceases to be found then goto step 1.
```

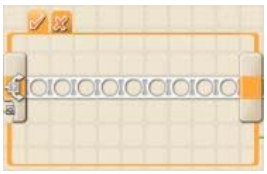
The program uses a single variable to maintain the arc length of the previous sweep. Each time a sweep finishes the variable gets doubled. This insures that the robot continues to trace out more of a circle both counter clockwise and clockwise until the line is found. Once the line is found the sweep variable gets reset to its initial value of two. This program takes advantage of MyBlocks to reduce the amount of clutter in the main routine. The routine to sweep clockwise and counter clockwise are each their own MyBlock. The routine to double the value in the sweep variable is also a MyBlock.

As with most line following routines, the performance of the routine in practice depends a great deal on the power settings for the motors. The setting that I found to work reasonably well with a new set of batteries is the 40% setting. In the former days of the RCX this type of program could only be accomplished using Not Quite C (NQC) or something similar. The NXT-G program shown below provides a good example of how much further we can go with the NXT-G programming environment.

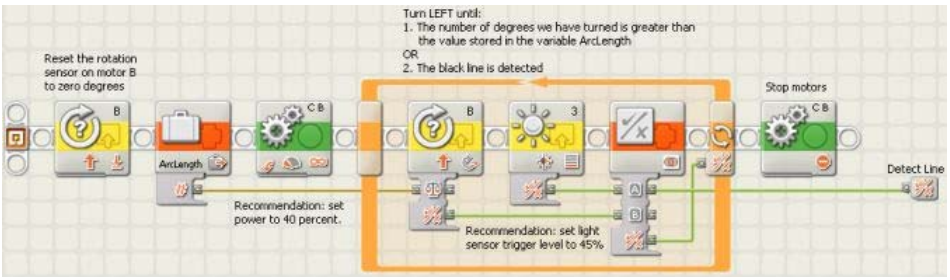
Incidentally, one really neat feature of the NXT-G programming environment is the abilility to print to HTML documents. The HTML documents are then easily edited to provide comments and explanatory information - such as this document. This document was produced by using the "Print to HTML" feature of the NXT-G programming environment. The additional text was added in a simple text editor.

Main Program

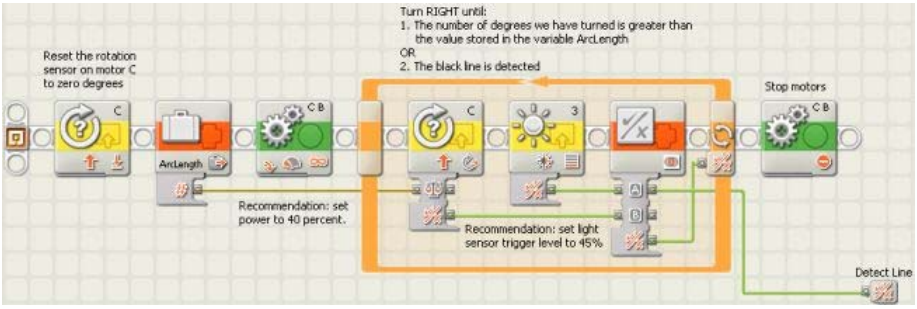




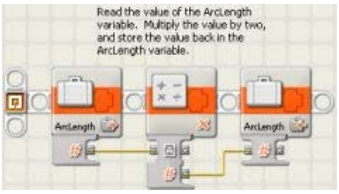
Left Sweep MyBlock



Right Sweep MyBlock



Double Sweep Arc MyBlock



References

- "Toeing the Line" - a very nice paper, by ninth grader Jonathan Gray, comparing several different line following strategies. Although written for the RCX, this paper provides an excellent example of performance testing and analysis, as well as presentation of results.

Source Code Files

- Line_Follower.zip

Address comments, suggestions or discussion to

xaos69@earthlink.net