

At all times, your bot should be doing one of these things:

1. Following a Line until it hits an intersection
2. Reading IR Input at an intersection
3. Navigating a Grid until it has “charged up” at the brightest spot and exited the grid
4. Reading a Barcode until it has read the specified number of ASCII character bytes
5. Checking for Obstacles on each path at an intersection
6. Basking in Glory at the end of the course

Following a line is the default mode, and your bot will be started on a line, facing the correct direction. Once it reaches an intersection, it should read the IR input to decide if it should switch modes:

* If the bot receives one of the commands 01 through 04 (left/right/forward/back), it should make the appropriate turn, if necessary, and continue Following a Line mode.
* If it receives one of the commands 05 through 08 (grid/barcode/distance/end), it should switch into another “mode” to complete the challenge given by that code. After completing the challenge, it should go back to Following a Line mode.
* Your bot should have some logic to decide which path to take if it doesn’t read any input within 5 seconds at an intersection.

If your bot reaches the end of a line without an intersection, it should turn around and go back.

Pseudocode for one possible implementation of this would be:

main()

while (1) {

if (intersection) {

readAndFollowInput();

intersection = false;

}

else {

intersection = keepFollowingLine();

}

}

readAndFollowInput()

command = readIRWithTimeout5Seconds();

if (command == TIMED\_OUT)

command = /\* whatever default behavior you want \*/

else if (command == LEFT) turnLeft();

else if (command == RIGHT) turnRight();

else if (command == FORWARD) {

do nothing;

}

else if (command == BACK) turnAround();

else if (command == GRID) navigateGrid();

else if (command == BARCODE) readBarcode();

else if (command == OBSTACLES) searchObstacles();

else if (command == END) celebrate();

// roll forward a bit, to get off the intersection/

// grid / end of the barcode

keepFollowingLine()

// This can be based on sample line follower or maze solver code

// Maze solvers might have logic for detecting intersections ;-)

// Also handle reaching the end of a line

searchObstacles()

// Rotate the bot to face down each possible path at this intersection,

// and check for any obstacles within distance specified

// Point the bot in the direction without obstacles, and start forward

celebrate()

// This should have an infinite loop, even if the bot does nothing,

// since the main() method of the bot is never supposed

// to exit while the bot is running