Toy Car Driven with Artificial Neural Networks and Neuroevolution

I want to create a system that teaches a toy car to navigate through human-designed tracks, such as mazes, to find a goal. The system will have multiple techniques to teach the car how to drive, one where the user does the teaching, and another where it teaches itself.

One will require a human to control the car through a training course, it will use the data it collects as input for a neural network. The user can then place the car in any environment; how well the car will be able to navigate through the environment will depend on how well the user has taught it.

Another mode of teaching will utilise 'Neuroevolution', which uses evolutionary algorithms to design an artificial neural network. The car will start off very incompetent at navigating a course, but over many generations it will self-learn the best actions to take when confronted with obstacles such as a left-turn or dead-end e.c.t. The fitness of each organism will be a value related to how close the car gets to the goal.

The car will have proximity sensors fitted all around it, the way it "sees" would be an array of values coming in from the sensors.

Ideally, I would like to have an actual physical toy car that the system controls. However, given that it would be easy to adapt from a virtual simulation to a physical car, I think I should start off with a simulation. And if I have the time, add support for a real-world toy car (either built by me, or purchased).

The software will be written in either C++ or Java. C++ would likely be better for working with a real toy car, however it would be harder. I'm more confident with Java, but it may be too high-level of a programming language for what I'm trying to achieve (for example, not being able to use pointer manipulation may make programming a neural network very challenging)