Automatic mobile robot control-Making a Route Follower

Pragna Das[1] and Lluís Ribas-Xirgo[1]

I. A Route Follower mobile robot

A. Part 1. Designing the Route Follower

The route follower is a program which will make the robot maneuver in a designated path. This module simulates a controller, typically embedded into a processor of mobile robot. It maneuvers the robot in a specific route in the simulated environment. The controller is based on an finite state machine and is programmed in LUA. The Figure 1 shows the given path in the environment. The distance in

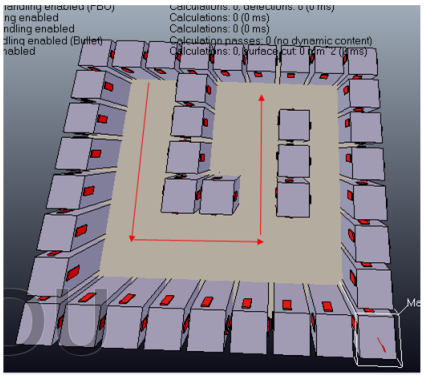

Fig. 1. The environment to maneuver the robot

the first step is 80 cm or 0.8 mts, the distance in second step is 75 cms or 0.75mts and the distance in the last step is 80 cms or 0.8 mts.

The Figure 2 shows the finite state machine.

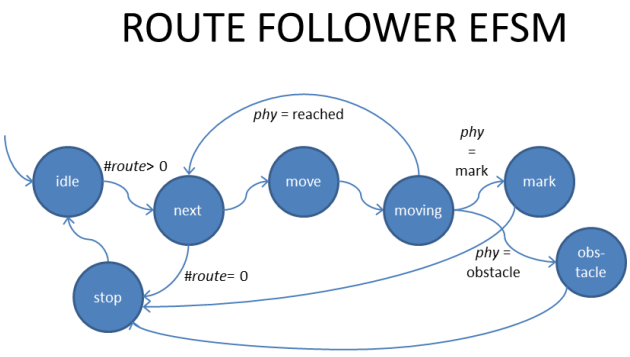

Fig. 2. Route Follower EFSM

*This work was done for Laboratory courses of embedded systems in UAB, Spain

[1]Pragna Das and Lluís Ribas-Xirgo are with Microelectronics and Systems Electronics Department, Autonomous University of Barcelona, Bellaterra, Barcelona, 08193 Spain {pragna.das,lluis.ribas}@uab.cat

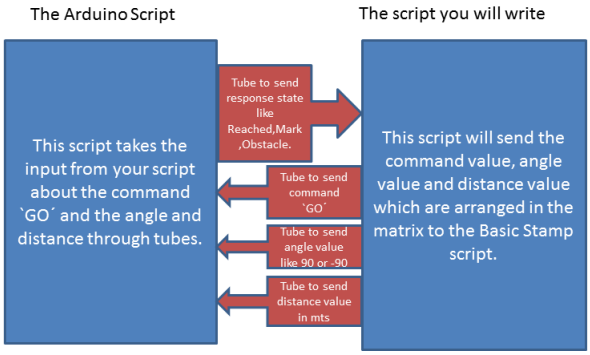

Fig. 3. The system of communication for route follower through tubes

B. Part 2. Programming the state machine

To make the robot move, the instruction string **GO** has to be given along with the <angle> it has to rotate and the ¡distance¿ it has to cover. This **GO** instruction has to be passed to the Arduino script along with the value of angle in numbers like 90 for right rotation and -90 for left rotation and the distance in mts. You will have to send the command **GO** and the angle and the distance value for each step of the route. So you can save the command, angle and distance in a 3 by 3 matrix, like **[[GO, 90, 0.8], [GO, -90,0.75], [GO, -90,0.8]]**. Also, the Arduino script will send response through tubes when rotation is complete and distance is reached. The responses are states like **Complete** and **Reached**. This two response will be given to your script again through tubes.(Refer to V-REP manual for Tube Communication [3]). You will use API functions like simTubeOpen(),simTubeRead() for this program. Please go through the API function lists for these API functions in the section Tube communication functionality [1]. The Figure 3 depicts how your script and the Arduino script will communicate. To follow the route given in Figure 1, the starting command can be **[[GO, 90, 0.8]**. Here, the command means to rotate first 90 degrees and then move ahead 0.8 mts. Similarly, you have to compute all the commands for the entire route. Then you can store all the commands in a matrix. The program of the state machine is described in Figure 4

1) The state machine can start with the state **IDLE** where you see whether any route is available and then go to the state **NEXT**. The variable *route* starts with the value of number of commands to be given to follow the route. The process of route following will come to an end when *route* will be equal to zero. In the process of route following, every time after sending each command, the value of *route* is reduced by 1.

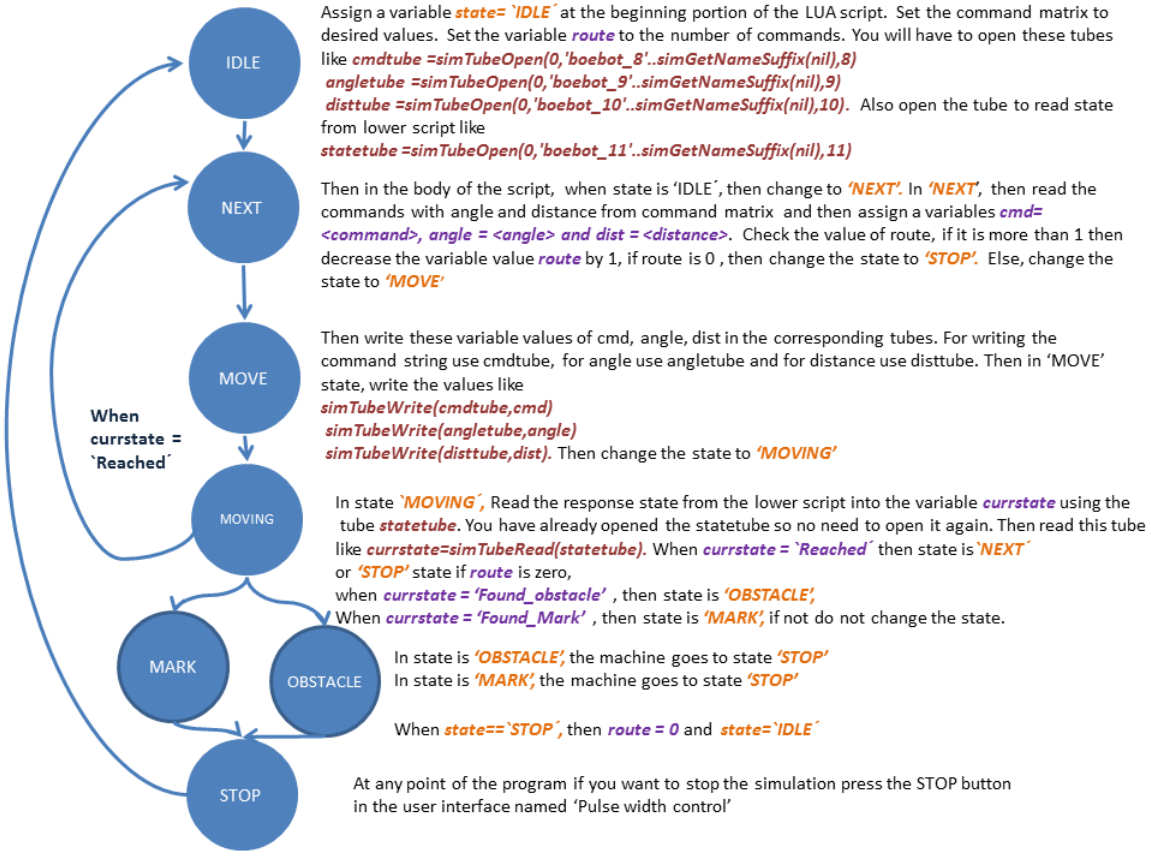

Fig. 4. The state diagram in detail

2) In the state **NEXT**, the commands from the matrix can be selected according to the route and the state should be changed to **MOVE**. Here the variable *route* can be reduced by 1
3) In the **MOVE** state, the command can be sent to the communication tube and state should be changed to **MOVING**
4) Now in the **MOVING** state, the response state from the Arduino script has to be read. If the response state is *Reached* then the machine checks the value of variable *route*, if it is zero, then machine goes to **STOP** state and if it is not, then it goes to **NEXT** state.
5) When the response in **MOVING** state is *Mark*, then the4 machine goes to **MARK** state and if the response is *Obstacle*, then the machine goes to **OBSTACLE** state
6) From both the states, **MARK** and **OBSTACLE**, the machine goes to **STOP** state

The LUA program is written in a non-threaded script [2] associated with the object RaspPI. This script will send commands to the Arduino script and this will also receive responses from Arduino script.

REFERENCES

[1] API functions for Tube Communications. http://www.coppeliarobotics.com/helpFiles/en/apiFunctionListCategory.html.
[2] Child Script. http://www.coppeliarobotics.com/helpFiles/en/childScripts.htm.
[3] Means of communication in and around V-REP . http://www.coppeliarobotics.com/helpFiles/en/meansOfCommunication.htm.