Obstacle detection and guiding the robot in the environment

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

I. Detection of obstacle for mobile robot

This simulation shows an obstacle detection module for the mobile robot.

A. Part 1. Obstacle Detection

The mobile robot which is simulated in V-REP is named NinjUAB and is equipped with an ultrasound sensor in the sonar module and have Infra-Red sensors on both sides. There are 2 Infra-Red sensors on each side of the NinjUAB, one on top and one on bottom. The sonar module is on the front of the robot. The ultrasound sensor is used to detect any obstacle in front of the NinjUAB and the Infra-Red sensors are used to detect the red mark on the boxes in the environment which signify the ports. The Arduino script controls the movement of the ultrasound sensor and also the four Infra-Red sensors. The sonar module which contains an ultrasound sensor in it can rotate full 180 and can give the position of the obstacle when detected in the range of the ultrasound sensor and in the proximity of the robot. Moreover, when the sensor is active, while the robot moves forward, the sensor can detect the obstacle in front without the necessity to rotate the sonar module. The robot moves around in the environment given in Figure 1. The state machine finds any obstacle present in the path given in the Figure 1 As seen in Figure 1, there are obstacles in the path of the movement of the robots. When the robot will encounter any of these obstacles, it will report to get further instructions.

B. Part 2. State Machine and Program for obstacle detection

For obstacle detection, a program is developed to maneuver the robot in the simulated environment in the designated path and when obstacle is found in the path, it will report. The Figure 2 depicts the state machine for this functionality. The process of obstacle detection is based on the state machine given in Figure 3

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**. You can have a variable *route* which will start 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, after sending each command, you can reduce the value of *route* by 1 every time
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 *Found_obstacle*, then the state machine should go to **NEXT** to find the alternate route. The alternate route can be stored in the command matrix as set of commands and these commands one by one can be again given to the robot as previously

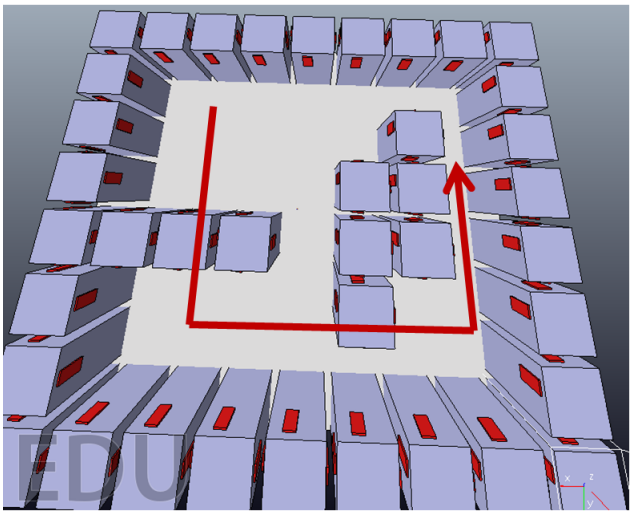

Fig. 1. The path the robot has to cover

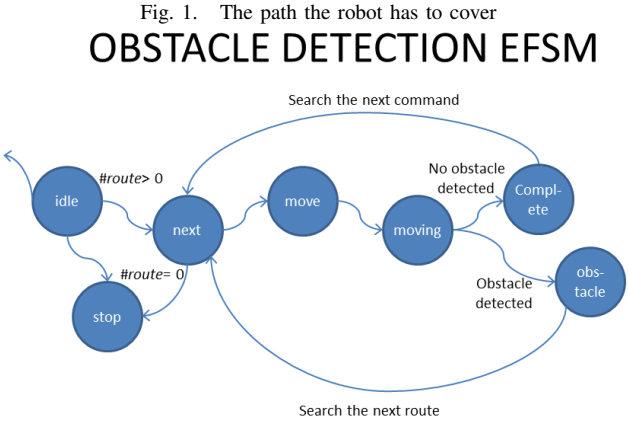

Fig. 2. The state machine to detect obstacles

*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

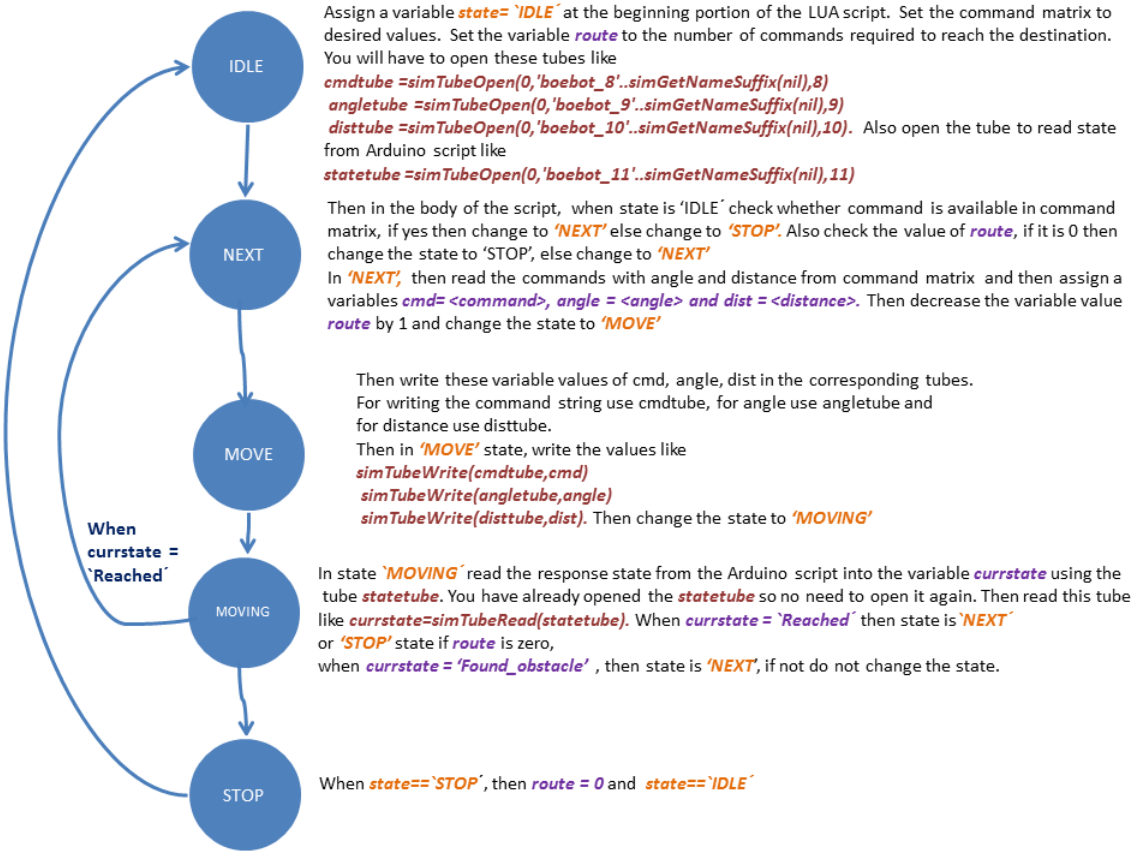

Fig. 3. The state diagram in detail

5) If in **MOVING**, the response 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
6) A sample route is given in the Figure 1 and the path has some obstacles in the middle
7) The goal is to detect these obstacle and get the next route to reach the destination point (the point where the red path ends)
8) The distance in the first step is 100 cm or 1 mt, the distance in second step is 100 cms or 1 mt and the distance in the last step is 60 cms or 0.6 mts
9) The route finally is a set of commands. Like the route given in Figure 1, the starting command can be **[[GO, 90, 1]**. Here, the command means to rotate first 90 degrees and then move ahead 1 mt. Similarly, all the commands for the entire route are computed and stored in a matrix
10) Also, here there is an obstacle in the given path, so an alternate route to maneuver the robot to the destination point is decided and saved in the matrix. This route will be used if the obstacle is detected in the path of the movement of the robot
11) The instruction string **GO** has to be given along with the ¡angle¿ it has to rotate and the ¡distance¿ it has to cover
12) 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
13) The command **GO** and the angle and the distance value for each step of the route is to be send to Arduino
14) So the command, angle and distance is saved in a 3 by 3 matrix, like **[[GO, 90, 0.8], [GO, -90,0.75], [GO, -90,0.8]]**
15) Also, the Arduino script will send response through tubes when rotation is complete and distance is reached
16) The responses are states like **Complete** or **Reached** or **Found_obstacle**
17) These responses will be given to your script again through tubes (Refer to VREP manual for Tube Communication [2])
18) The Figure 4 depicts how your script and the Arduino script will communicate.
19) The state machine is translated into a LUA script (non-threaded child script [1]) which is associated with the RaspPI .
20) The robot goes from one state to another and move in an automated way in the given path. Please note, the commands to move the slider of the user interface is given in the script of the Arduino and you dont need to change that
21) Each and every script of the scene is run once in each simulation time step 3

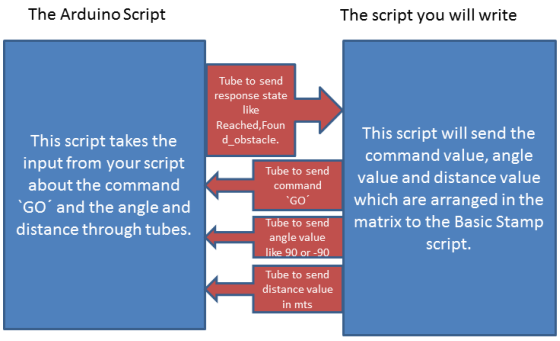

Fig. 4. The system of communication for obstacle detection through tubes

REFERENCES

[1] Child Script. http://www.coppeliarobotics.com/helpFiles/en/childScripts.htm.
[2] Means of communication in and around V-REP . http://www.coppeliarobotics.com/helpFiles/en/meansOfCommunication.htm.