A MOVEMENT CONTROL OF A SMALL MOBILE 3-PI ROBOT IN A MAZE USING ARTIFICIAL NEURAL NETWORK

Dušan Horváth[1], Zuzana Červeňanská[1], Rastislav Ďuriš[1]
[1]Faculty of Material Science and Technology,
Slovak University of Technology in Bratislava, Trnava, Slovak Republic
E-mails: dusan.horvath@stuba.sk, zuzana.cervenanska@stuba.sk, rastislav.duris@stuba.sk

Abstract – An intention of this article is to present a technical implementation of control of a small mobile 3-Pi robot which moving in a maze along a predefined guide line. To acquire the robotic car position information, the reflectance infra-red sensors serve as input sensors. The control of the direction of the robot's movement is performed by a single-layer neural network including two neurons. The weights (memory) of the neurons are adapting with respect to input sensors signals and the output calculated via Hebbian learning. Based on previous experiments, activation function bipolar sigmoid performs the best for a maze solving problem.

Keywords: Mobile Robot, Control, Neural Network, Maze Solving Robotic Vehicle, Navigation.

1. Introduction

A movement control of autonomous vehicles and mobile robots is in the centre of attention for many years. An applicability of these equipments in service and exploration activities or in conditions unsuitable or dangerous for human demands to ensure reliable movement of the mobile robot according to navigation requirements.

As for the movement direction along the guide line controlled by the neural network, it is necessary to process the data from the input sensors quickly and as accurately as possible, with the smallest possible deviation of the calculated data from the expected data. For this reason, the aim is to achieve the smallest possible learning error when learning neurons so that the robot can follow the guide line as accurately as possible.

2. Movement Control Methods for Mobile Robots

Nowadays, the robot control subsystem can be implemented by software using the main following methods:

- by evaluating the data from the input sensors based on various software conditions (if, switch), by means of which the input data (input vector) from the robot sensors are assigned the required outputs (e.g. motors control);
- using PID controllers, where the input data (input vectors) are compared to the expected value and the difference of these values determines the size of the feedback control e.g. motors [1–3] – see Figure 1;
- applying a neural network (NN) [1,2,4]. So-called weights (neuron memory) are calculated depending on the data from the input sensors (input vector) and the expected outputs in the process of neurons learning. During robot operation, the control subsystem calculates the robot's response to the input vectors based on the weights and input vector. The mathematical model of NN is explained in more detail in text that follows.

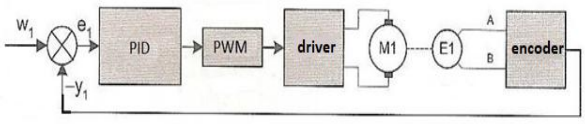

Figure 1: An example of electric motor control using a PID controller, adapted from [1].

Navigation techniques that deal with specific problems of path planning and optimization in a complex environment, such as different Artificial Intelligence methods, fuzzy logic systems, Neural Networks (NN), neuro-fuzzy and computing techniques like genetic algorithms, nature-inspired computing technique, Ant Colony Optimization, and Particle Swarm Optimization, are mentioned in a review article [4].

Monitoring the robot position includes an applying an appropriate sensing method. A comprehensive overview of sensors classification, their characteristics and utilization of sonar, laser and infrared sensors for detecting position can be found in [2]. Reflectance infrared sensors or single-line video camera are used to monitor the guide line.

The method of creating a map of the environment is also implemented for the faster movement along

the guide line [2–5]. Motion subsystem sensors – encoders – are used to calculate the travelled distance and sensors of the robot's motion subsystem navigate the robot.

The most common method of navigation having its place among the relative navigation methods is odometry [1–3]. The robot updates its position and records it on a map of the environment. Using the environment map, e.g. the robotic car is able to move very quickly along the guide line. Figure 2 depicts the brief scheme of navigation process based on localization and mapping [2].

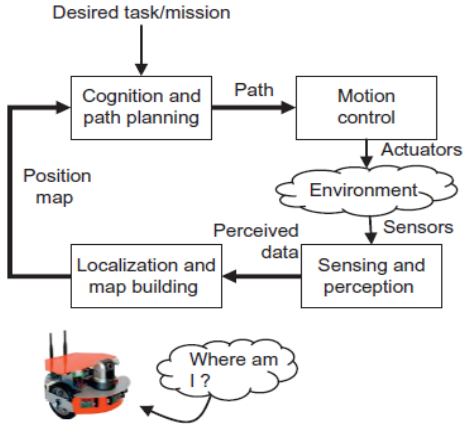

Figure 2: The background concept of navigation via map-based localization [2].

Several well-known maze exploration algorithms are briefly described in [6]. There is also presented an implementation of a maze solving robotic vehicle built on Arduino UNO platform, equipped with three ultrasonic sensors, two electro-mechanical encoders, and a motion tracking device consisted of 3-axis accelerometer and 3-axis gyroscope for detection of location in the maze.

3. Implementation of Hebbian learning in NN

Principal navigation problems such as recognition, adaptation and action are very similar to cognitive tasks of human brain and an artificial neural network representing a massive system of parallel distributed processing elements (neurons) linked in a graph topology can deal with them. After learning, the neural network can express the knowledge implicitly in the weights associated to the inputs of neuron [7].

In general, the neuron activity can be modelled as function of n input variables (input vectors $\mathbf{x}_j = [x_1, x_2, \ldots, x_i, \ldots, x_n]^T$) which transform m vectors $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_m]$ to the output vector $\mathbf{Y} = [y_1, y_2, \ldots, y_j, \ldots, y_m]$ with binary value elements.

In addition, an input that is connected to value 1 with the weight called bias (named b), is assigned to the neuron. Value b is usually related with a postsynaptic neuron threshold ($b = -\theta$).

The weighted linear combination of the inputs (1) represents an overall stimulus – a neuron potential. The weights w_i carry information on network learning process which is based on a sequential adaptation of neuron weights.

$$z_j = \sum_{i=1}^{n} x_i w_i + b \tag{1}$$

Finally, the activation function f (e.g. Sgn(z_j) or sigmoid functions) converts the inner neuron potential (1) to output expressed in the form (2).

$$y_j = f\left(\sum_{i=1}^{n} x_i w_i + b \right) \tag{2}$$

Considering Hebbian learning rule, for more than one neuron a new weight w_{ij} (t+1) is changed with respect to the expression (3), where x_i and y_j are the output values of neurons i and j, respectively, which are connected by the synapse w_{ij}, and η is the learning rate (x_i is the input to the synapse) [7].

$$w_{ij}(t+1) = w_{ij}(t) + \eta . x_i(t) . y_j(t) \tag{3}$$

If the network includes only a single neuron, the weight w_i is adapted in the form (4) and bias b with respect to (5).

$$w_i(t+1) = w_i(t) + \eta . \Delta . x_i(t) \tag{4}$$

$$b_i(t+1) = b(t) + \eta . \Delta \ , \tag{5}$$

Where:

$$\Delta = o(t) - y(t) \tag{6}$$

The value Δ corresponds to the difference (6) between the expected network response $o(t)$ and its actual calculated response $y(t)$ [1].

Hebbian learning takes part successfully in learning the simplest single-layer neural networks that linearly separate input vectors of a training set into two classes. An example of design of neural network aimed to such type of classification [1] with respect to the network structure and its learning is described below. A typical structure of a single-layer single-neuron network with two inputs and logical function AND is shown in Figure 3.

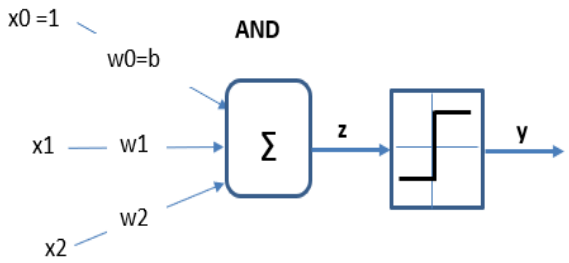

Figure 3: An illustration of a single-neuron network structure for logical AND function

In a first step, all inputs are summed by expression (1). The resulted value z_j enters as an argument to the activation function – the unit step function in this case. The output o as an expected network response to inputs x_1, x_2 meets the truth table of logical operation AND shown in Table 1. The actual calculation of weights (an adaptation) and bias takes place until the Δ value is zero for all input vectors x_i (there are four ones in our example in Table 1: $\boldsymbol{x}_1$ = [0,0], $\boldsymbol{x}_2$ = [0,1], $\boldsymbol{x}_3$ = [1,0] $\boldsymbol{x}_4$ = [1,1]).

Table 1. Input vectors and expected outputs of the network training set for logical operation AND

AND		
x_1	x_2	o
0	0	0
1	0	0
0	1	0
1	1	1

Regarding a geometric point of view, completed learning process means finding a hypersurface in a space of input vectors that separates individual classes [1].

The analytical form of hypersurface is determined by equation (7).

$$\sum_{i=1}^{n} x_i w_i + b = 0 \tag{7}$$

If we consider two input vectors, the hypersurface represents a line in the plane, perpendicular to the weight vector. Calculated values for an identification line which determines a linear separability of space applying weights and bias w = [0.7, 0.3], b = 0.9 are x_1 = 0.9/0.7 = 1.28, x_2 =0.9/0.3 = = 3. These values determine the axis coordinates for both of points belonging the straight line which linearly separates plane points into two classes (see Figure 4). The class "0" means inhibition of neuron activity and class "1" its activation.

The whole learning process of neuron according to logical operation AND is presented in Table 2. In the last four rows of Table 2, it can be seen that value of learning error delta Δ equals zero for all combinations of input vectors and expected outputs. Network learning is completed. The result of the value calculation is one of the possible, because the initialization of the weights and bias is random: w = [0.2, 0.3], b = 0.1.

Table 2. The adaptation process of weights calculation for AND operation

Logical function AND

line	x[0]	w[0]=b	x[1]	w[1]	x[2]	w[2]	output	z	y	delta Δ
1	1	0.1	1	0.2	1	0.3	1	0.6	1	0
2	1	0.1	0	0.2	1	0.3	0	0.4	1	-1
3	1	-0.4	1	0.2	0	-0.2	0	-0.2	0	0
4	1	-0.4	0	0.2	0	-0.2	0	-0.4	0	0
5	1	-0.4	1	0.2	1	-0.2	1	-0.4	0	1
6	1	0.1	0	0.7	1	0.3	0	0.4	1	-1
7	1	-0.4	1	0.7	0	-0.2	0	0.3	1	-1
8	1	-0.9	0	0.2	0	-0.2	0	-0.9	0	0
9	1	-0.9	1	0.2	1	-0.2	1	-0.9	0	1
10	1	-0.4	0	0.7	1	0.3	0	-0.1	0	0
11	1	-0.4	1	0.7	0	0.3	0	0.3	1	-1
12	1	-0.9	0	0.2	0	0.3	0	-0.9	0	0
13	1	-0.9	1	0.2	1	0.3	1	-0.4	0	1
14	1	-0.4	0	0.7	1	0.8	0	0.4	1	-1
15	1	-0.9	1	0.7	0	0.3	0	-0.2	0	0
16	1	-0.9	0	0.7	0	0.3	0	-0.9	0	0
17	1	-0.9	1	0.7	1	0.3	1	0.1	1	0
18	1	-0.9	0	0.7	1	0.3	0	-0.6	0	0
19	1	-0.9	1	0.7	0	0.3	0	-0.2	0	0
20	1	-0.9	0	0.7	0	0.3	0	-0.9	0	0

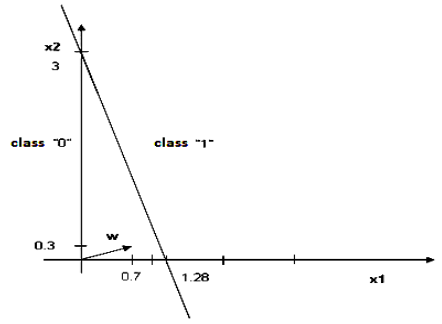

Figure 4: Division of the plane into classes "0" and "1" according to the results in Table 2

4. Applying a Neural Network for the Movement of a 3-Pi Robotic Car Control using Activation Function Bipolar Sigmoid

The scheme in Figure 5 presents an applied single-layer network consisted of two neurons, which is used to control the 3-Pi robot motion in the maze. One neuron is reserved for one motor. The neural network has five inputs. The inputs are generated by infrared sensors for sensing the black guide line.

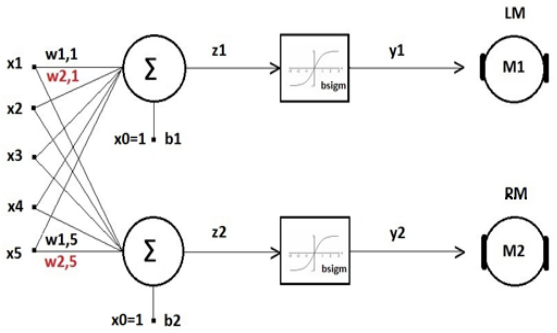

Figure 5: The scheme of applied neural network with activation function bipolar sigmoid

Five inputs for two neurons allowed 32 combinations of zeros and ones for possible network inputs. Actually, the sufficient number for well-defined robotic car movement in the maze was ten combinations. Therefore, we designed a table of only ten input vectors and an expected robot motion (Figure 6,7) for the bipolar sigmoid (hyperbolic tangential [2]) activation function (8).

$$f(x)=\frac{1-exp(-x)}{1+exp(-x)} \tag{8}$$

This function controlled of motor turning in two directions – forward and back. The value –1 ensured back direction of an engine rotation, the value 1 forward direction, respectively. All calculated values of weights and bias entered into the robot control subsystem. The weight coefficients $w_{1,1}$ to $w_{1,5}$ were the weights for the individual inputs of neuron which controlled the left motor (LM). Similarly, $w_{2,1}$ to $w_{2,5}$ were the weights for the individual inputs of neuron for the right motor control (RM).

The proposed input vectors were sufficient as the training set for the movement of the 3-Pi robot in the maze. With a large number of input vectors, it is not possible to use Hebbian learning because the training set is not linearly separable. In this case, you need to use a multi-layered network such as e.g. Back-Propagation type.

The results of calculation of the weights (neurons memory) for each neural network neuron with bipolar sigmoid activation function are presented in Figure 6. Besides the weights, there are also input vectors with expected and calculated values.

The letter "*o*" indicates the expected value at the output of individual neurons and the letter "*y*" represents the calculated value at the output of individual neurons.

The letter "*d* "indicates a learning error Δ (delta). Because the learning error Δ equals 0.001, the calculated weight coefficients will provide practically the same value at the output of the individual neurons as the expected values.

Left motor
Number of neurons=1
Number of inputs for one neuron=5
Number of input vectors=10
Learning rate=0.500000
delta_err=0.010000
Ini subor=labyrint-LM-bsigm8.ini
Activation function - bipolar sigmoid
w[5]=-42.214 w[4]= 5.492 w[3]=11.766 w[2]=11.763 w[1]=11.762 w[0]=-5.295
00000 o=-1.00 y=-0.99 d=-0.01
00100 o= 1.00 y= 1.00 d= 0.00
01000 o= 0.10 y= 0.10 d= 0.00
00010 o= 1.00 y= 1.00 d= 0.00
11100 o=-1.00 y=-1.00 d=-0.00
10000 o=-1.00 y=-1.00 d= 0.00
11111 o=-1.00 y=-1.00 d=-0.00
10001 o=-1.00 y=-1.00 d=-0.00
00111 o= 1.00 y= 1.00 d= 0.00
00001 o= 1.00 y= 1.00 d= 0.00
Right motor
Number of neurons=1
Number of inputs for one neuron=5
Number of input vectors=10
Learning rate=0.500000
delta_err=0.010000
Ini subor=labyrint-PM-bsigm8.ini
Activation function - bipolar sigmoid
w[5]=11.238 w[4]= 9.126 w[3]= 1.950 w[2]=-5.261 w[1]=-10.760 w[0]= 5.466
00000 o= 1.00 y= 0.99 d= 0.01
00100 o= 1.00 y= 1.00 d= 0.00
01000 o= 1.00 y= 1.00 d= 0.00
00010 o= 0.10 y= 0.10 d=-0.00
11100 o= 1.00 y= 1.00 d= 0.00
10000 o= 1.00 y= 1.00 d= 0.00
11111 o= 1.00 y= 1.00 d= 0.00
10001 o= 1.00 y= 0.99 d= 0.01
00111 o=-1.00 y=-1.00 d=-0.00
00001 o=-1.00 y=-0.99 d=-0.01

Figure 6: Input vectors and calculated weights for 3-Pi robotic car motion in the maze

Figure 7 shows the values of ten input vectors transmitted by five infrared sensors. The set of input vectors represented the training set for neural network.

The white area was evaluated as logical 0 by infrared sensors, the black area as logical 1, respectively.

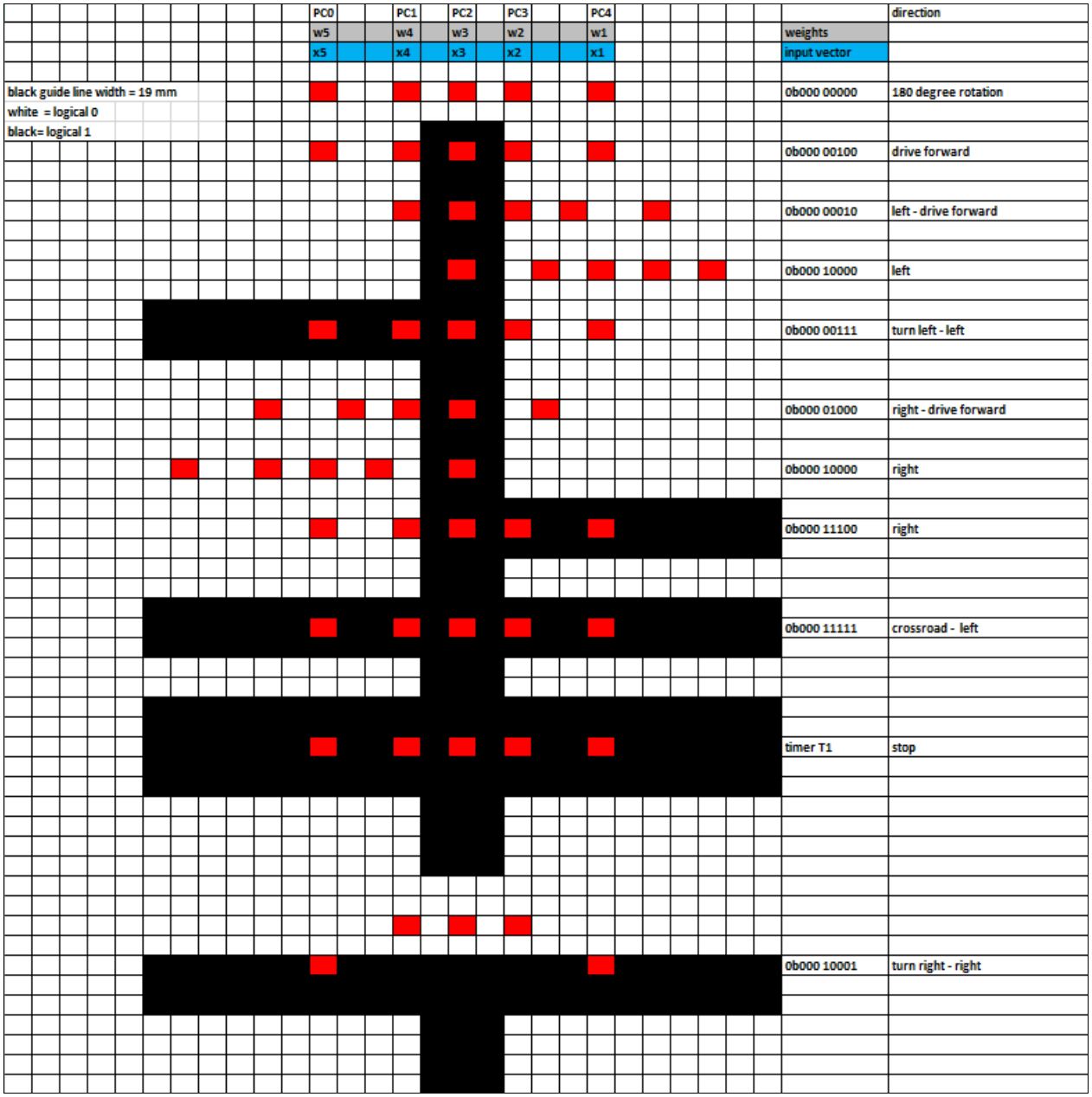

Figure 7: Applied input vectors for the robotic car movement along the line

5. Technical Implementation of the Robotic Car Motion in the Maze and Results

The shape of the tested maze used in experiment is depicted in Figure 8. The beginning of the track is marked "ŠTART" and the end of the maze is marked with a wide transverse black strip.

The robot goes through the whole maze until it finds the end (exit) of the maze.

The robot stops on a wide transverse black tape. We used a 3-Pi two-wheeled robotic car by Pololu [8] (see Figure 9 and Figure 10 for the top and bottom view) to verify the functionality of a designed neural network with a bipolar sigmoid activation function. Five reflectance infra-sensors were used to sense the black guide line. An 8-bit microcontroller ATMEGA328P by ATMEL [8] controlled the movement of the 3-Pi robotic car. All main components can be seen from the bottom view in Figure 10.

As it moves along the black line, the robotic car detects the reflection of infrared light from the area below it with its infrared sensors. Because the reflection of infrared light from a black surface is different from that of a white surface, a different voltage appears at the output of the infrared sensors. The voltages from the infrared sensors were digitized, and they represented the input vectors into the table of the expected movement of the mobile robot.

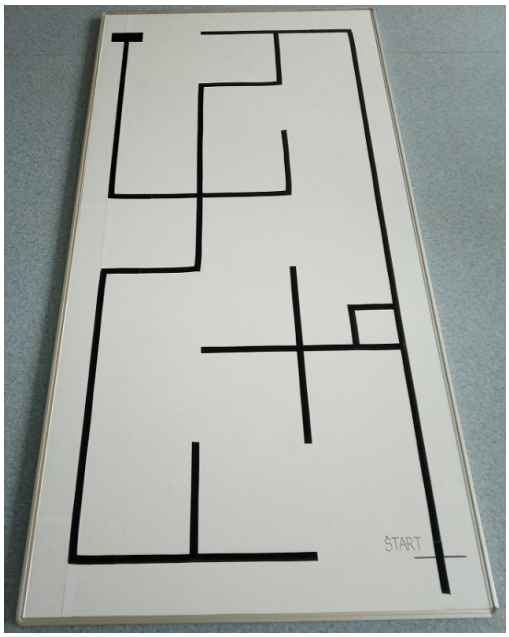

Figure 8: The maze

The input vector and weights were processed by the expression (1). The resulted values z_j entered directly to the robot's motion subsystem. The response of neural network to input vector **X** was y_j, and the motor speed was y_j multiplied by maximal motor speed then. The result of this activity was the movement of the robot along the path.

As for testing applying different activation functions, their comparing showed that neural network with linear and sigmoid activation functions in the role of control element did not perform correctly – the robotic car did not find the path through the maze.

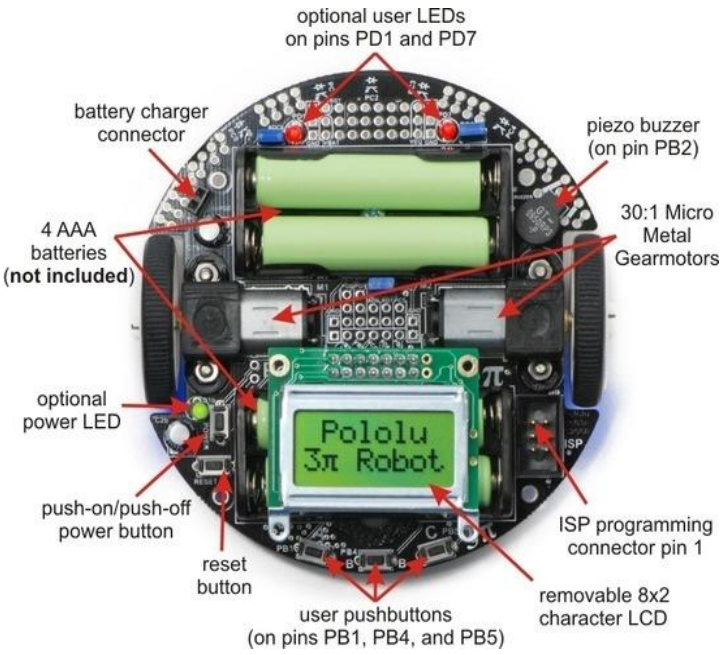

Figure 9: Top view of a 3-Pi robotic car [8]

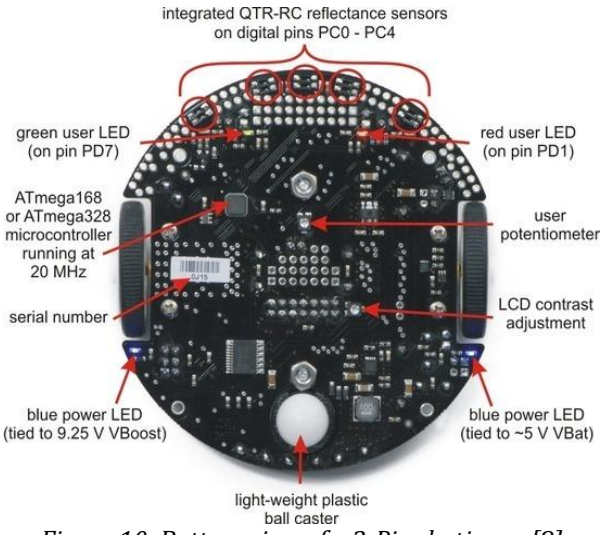

Figure 10: Bottom view of a 3-Pi robotic car [8]

Figures 11 and 12 depict the expected and calculated values of network response to ten input vectors using Hebbian learning with a learning error $\Delta = 0.001$ and learning rate $\eta = 0.5$ for the individual motors. Applying the calculated values in the control of both of motors led to the smooth and precise movement of the robotic car along the guide line in the tested maze.

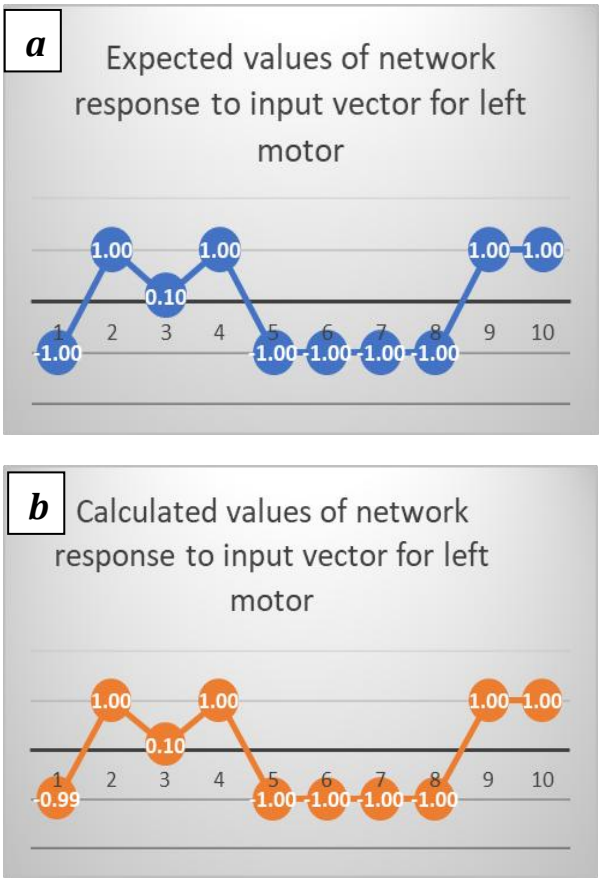

Figure 11: Comparison of expected (a) and calculated (b) values for the left motor of the robotic car

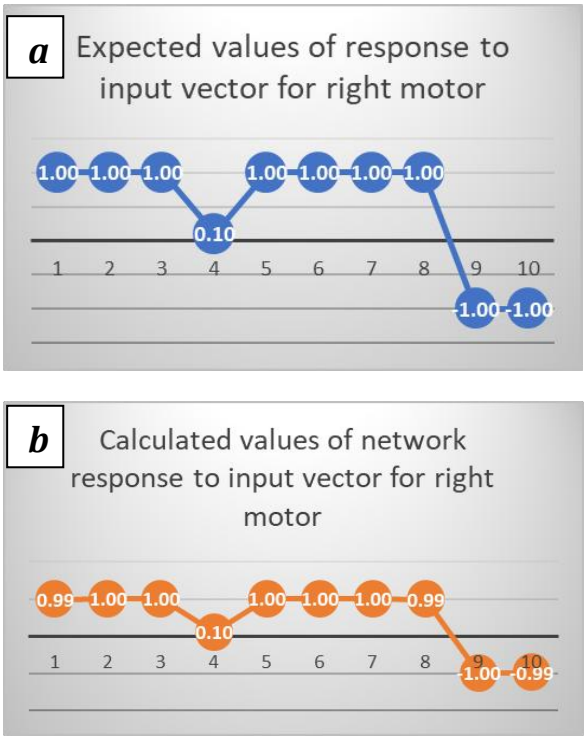

Figure 12: Comparison of expected (a) and calculated (b) values for the right motor of the robotic car

6. Conclusions

The proposed single-layer neural network consisted of two neurons with five inputs for both of them and bipolar sigmoid activation function works well for the solving movement of a 3-Pi robot in the maze or along a circular or oval path determined by guide line too. Experiments with different activation functions showed that the best choice of activation function for implementation in this type of control is bipolar sigmoid function when moving a mobile robot in the maze.

Except for a simple technical solution of the maze problem, the demonstration of robotic car movement in the maze can assist as a good motivation element for educational purposes. All obtained results and the simple technical implementation suit in subjects focused to neural networks theory and its application in the field of mobile robot control.

For the movement of a robotic car on more complex tracks, where the control subsystem of the robotic car has to solve not only curves, but also to reduce the speed in curves so that the car does not run out of track, retarders, intersections and hills, it is necessary to use multilayer networks.

For multi-layer networks with a larger number of neurons in the layers and a larger number of inputs, it is not possible to manually define the parameters of such a network and therefore some of the existing computational algorithms are used, the best known is the "Back-Propagation" algorithm. It is a two-layer or multi-layer network formed by neurons with a differentiable activation function. The sigmoid function in the form $y=1/(1+e^{-\lambda z})$ is most often used as an activation function in this case. Future interest and the direction of experiments with driving a robotic car via a neural network will be connected to the use of such a multi-layer network for navigation in a more complex environment.

Acknowledgements

This work was supported by Slovak Cultural and Educational Grant Agency KEGA within the project 029STU-4/2018.

References

[1] Novák, P. Mobilní roboty (1. diel). Mobile robots (1st part). 2005, Ben- technical literature, Praha.
[2] Tzafestas, S. Introduction to Mobile Robot Control (1st edition). 2014, Elsevier
[3] Siegwart R., Nourbakhsh, I.R. Introduction to Autonomous Mobile Robots. 2004, Massachusetts Institute of Technology
[4] Gul, F., Rahiman, W., Alhady, S.S. l. N. A comprehensive study for robot navigation techniques. 2019, Cogent Engineering, 6:1
[5] Kleeman, L. Advanced sonar and odometry error modeling for simultaneous localization and map building. 2004, Proceedings of the 2004 IEEE/RSJ international conference on intelligent robots and systems, Sendai, Japan, pp. 1866-71
[6] Bienias, Ł., Szczepa´n ski, K., Duch, P. Maze exploration algorithm for small mobile platforms. 2016, Image Processing & Communications. 21:3, pp. 15-26
[7] Janglová, D. Neural Networks in Mobile Robot Motion. 2004, International Journal of Advanced Robotic Systems, 1:1, pp. 15-22
[8] Pololu 3pi Robot User's Guide. www.pololu.com/docs/0J26/all. accessed at: 23.09.2020