Middle-East Journal of Scientific Research 24 (S1): 124-130, 2016
ISSN 1990-9233
© IDOSI Publications, 2016
DOI: 10.5829/idosi.mejsr.2016.24.S1.27

An Intelligent Navigator for Indoor Applications Using Raspberry PI

M. Suganya, S. Jayanthy and N. Geraldine Shirley

M.E., Embedded System Technologies, Department of ECE, SREC, Coimbatore, India

Abstract: This paper proposes an intelligent mobile robot, which can navigate to the destination with position stabilization and obstacle avoidance techniques using fuzzy neural networks. The wheeled mobile robot integrates an RFID reader to determine its location by reading passive tags placed in the structured environment and an ultrasonic sensor to find out obstacles in the path. Standalone activity of the robot is achieved by a digital computer; Raspberry Pi. In this proposed system, Fuzzy Neural Network has been developed to discover the path for the designated wheeled robot. The developed FNN uses the fuzzy set instead of a set of membership values and it is robust in dynamic environment, the performance of the robot behavior can be substantially improved in the presence of obstacles on a structured environment. The validity and the advantages of the proposed method are shown by experimental results, which are using the designated wheeled mobile robot with the provided position information.

Key words: Fuzzy neural network (FNN) · Obstacle avoidance · Position stabilization · Structured environment · Wheeled mobile robots

INTRODUCTION

The wheeled mobile robot structure is an active research area. For example, numerical methods for trajectory path tracking, positioning, navigation and path separation have been proposed. In the control of the wheeled mobile robots, there are two approaches[1]: the deliberative approach and the reactive approach. Whereas the path deciding, trajectory path tracking and positioning belong to the deliberative approach, navigation, path separation and obstacle avoidance belong to the reactive approach. The deliberative approach creates exact planning so as to accomplish the task by using the detailed geometric model of the environment as well as the precise theory.

This approach is[2], however, robust in the uncertain environment in practice because when the surrounding environment has changed, the approach should involve the recalculation of the planning in order to accomplish the task. This drawback becomes crucial in the case of the mobile robots because the environment surrounding the mobile robot can keep changing so rapidly. In addition, since the computational complexity using the deliberative approach becomes quite large, the control effort also becomes large accordingly. On the other hand, the reactive approach works in such a way that as soon as the new information comes from the sensors, the actuator is made to react to it instantly so as to accomplish the task. Since the reactive approach does not need to know the whole planning of the task beforehand and just needs to react against the changing environment, it generally requires much less computation effort and spends much shorter time to achieve the [3]desired behavior compared with the deliberative approach. Therefore, a lot of research work has been based on the reactive approach in mobile robot control.

In the reactive approach, fuzzy methods are widely used because they are capable of making inferences even for uncertainties. We should consider the various situations such as the uncertain or unstructured environments in the control of mobile robots.

It is difficult and time consuming [4]for human experts to examine all the input–output data from a mobile robot to find a suitable behavior, since there are huge number of situations, when moving the mobile robots. Thus, intelligent mobile robots with fuzzy methods have been employed to cope with this difficulty.

Corresponding Author: M. Suganya, M.E., Embedded System Technologies, Department of ECE, SREC, Coimbatore, India.

System Design: The proposed system is developed with a standalone processor; Raspberry pi to achieve fuzzy neural schema and independent wheeled computer. The wheeled robot integrates a power supply, ultrasonic sensor and a RFID reader. Fuzzy Neural Network is designed in order to find various[5] paths and to choose the optimum path between the current and designated co-ordinates. Ultrasonic sensor is used to find out the obstacles in the path when the wheeled robot is moving in its path. RFID reader is used to read the Passive tags, which are embed with the structured environment. Each passive tag is assigned with a co-ordinate, when the reader reads the passive tag it [6]will identify its current co-ordinate then with the help of FNN it will calculate the optimum distance, angle and navigate through the desired path.

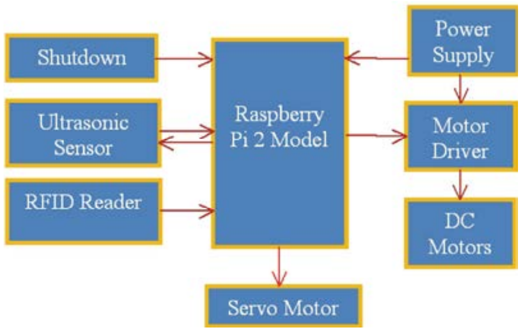

Fig. 1: Block Diagram

The proposed wheeled robot comprises of a processor named Raspberry Pi 2 as shown in Fig.1. The processing unit is integrated with Broadcom's BCM2836; an ARM7 architecture based Processor. The RFID reader and passive RFID tags plays an important role in finding out the various paths. FNN is created in such a way to explore the shortest optimistic path among the various paths[7]. The optimum path is used by the wheeled robot to reach the respective designation. The dynamic obstacle in the optimistic path is detected by using ultrasonic sensor. The sensing range of the sensor is to be set by the developer, depending upon the prototype model. In this proposed system, the sensitivity is set to 20cm to detect the obstacles. The wheeled robot is navigating in its path until it detects the obstacle, if any hurdle is detected the tilt angle of ultrasonic is carried out through the servomotor based on FNN. Then the wheeled robot navigates to the next nearby co-ordinate and determines the optimum path from that co-ordinate. After determining path, the wheeled robot proceeds further as explained in the following pseudo code.

Pseudo Code:

```
• import header files
• declare variables and pin ins/outs
• initiate shutdown trigger with high priority
• initiate scan mode of raid reader
• store 2- 2D arrays for turgid vs. distance, turgid vs. direction
• get the target co-ordinates from the user as a(k, l)
• def obstacle_ avoidance ()
• initiate ultrasonic sensor
• if distance < 20 cm
• then move backward
• by changing servos duty cycle measure left and right distances in
  multiples of 15degrees until left or right distance greater than 20cm
•         if left_ dist < right_ dist
•             then move right corresponding with servos degree
•         else move left corresponding with servos degree
•     else move forward
• navigate the navigator as per obstacle_ avoidance function until it
  scans the first passive tag (each passive tag is associated with a 2-D
  coordinate as a(m, n))
• compare current location and target and find the distance and direction
• while (k!=m || l!=n)
•     if(k == m)
•         if(l>n)
•             Turn right 90degrees and move forward until any
              obstacle detected or l==n
•         if(l<n)
•             Turn left 90degrees and move forward until any obstacle
              detected or l==n
•     if (k<m)
•         if (l==n)
•             move forward until any obstacle detected or k==m
•         else calculate the distance and angle between the source and
          destination
•             if (l<n)
•                 turn right for the calculated angle then move
                  forward for the calculated distance until any obstacle
                  is detected
•             if (l>n)
•                 turn right up to (180-calculated angle) degree then
                  move forward for the calculated distance until any
                  obstacle is detected
•     if(k>m)
•         if (l==n)
•             Turn 180degree and move forward until any obstacle
              detected or k==m
•         else calculate the distance and angle between the source and
          destination
•             if (l>n)
•                 turn left up to(180-calculated angle) then move
                  forward for the calculated distance until any obstacle
                  is detected
•             if (l>n)
•                 turn left for the calculated angle then move forward
                  for the calculated distance until any obstacle is
                  detected
•     if obstacle identified
•         then move according to obstacle_ avoidance function and use
          Pythagoras theorem to calculate distance and direction
```

Wheeled Mobile Robot Model: The robot posture as shown in Fig. 1 in the Cartesian coordinates is represented by three variables as

$$p = [x, y, \theta]^T \tag{1}$$

Where x and y are the coordinates of the center of mass of the robot and its heading direction angle θ is taken counterclockwise from the X-axis. The mobile robot can be controlled by

$$r = [v, w]^T \tag{2}$$

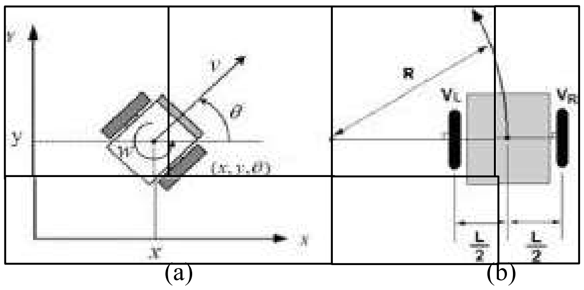

Fig. 2: Kinematic model and rotation radius of the mobile robot.

Where v is a linear velocity and w is an angular velocity. Then, the kinematic model of the mobile robot shown in Fig. 2(a) is described by the following equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{3}$$

The linear and angular velocities are related by each wheel velocity as

$$V = \frac{V_R + V_L}{2} \tag{4}$$

$$W = \frac{V_R - V_L}{2} \tag{5}$$

Where v_R and v_L are right and left wheel velocities, respectively and L is the distance between wheels. Once the linear and angular velocities are decided, the rotation radius of the mobile robot in Fig. 2(b) is also decided and using the relationship

$$\frac{v_L}{R - \frac{L}{2}} = \frac{v_R}{R + \frac{L}{2}} \tag{6}$$

the rotation radius of robot can be calculated as

$$R = \frac{L(v_R + v_L)}{2(v_R - v_L)} = \frac{v}{w} \tag{7}$$

From (7), it can be inferred that if the linear or angular velocity is saturated, the mobile robot may not move toward the goal position smoothly because of the saturated linear or angular velocity, which will be confirmed later in the simulation and experimental results.

Type-1 Fuzzy Neural Network For Obstacle Avoidance In The Position Stabilization: Since the proposed system is modified based on the Type-1 Fuzzy Neural Network (T1FNN) to achieve the improved performance, it is necessary to describe the T1FNN first in more detail than is done, which is an earlier conference version of this work. Whereas Hagias used the hierarchical structure, T1FNN uses the neural network method to combine the multiple input and output data[8]. T1FNN receives the necessary input data from the real-time system and then generates the desired linear and angular velocities to achieve both obstacle avoidance and position stabilization. As shown in Fig. 3, T1FNN consists of three layers of neural network such as input, hidden and output layers and each layer is connected via the appropriate weights.

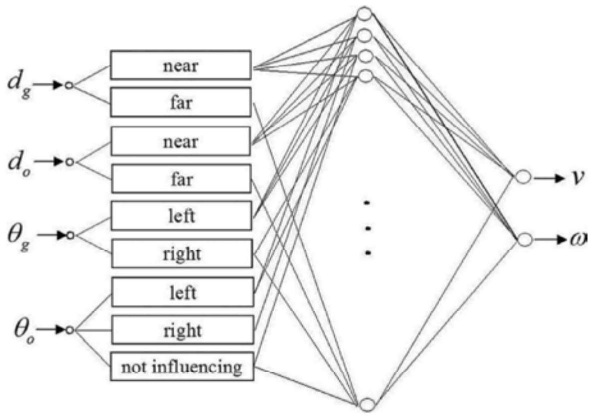

Fig. 3: Structure of the T1FNN.

The role of each layer can be explained as follows[9]. The input layer handles the input data as linguistic terms to process the fuzzification and then, it calculates the membership values using the membership functions. The calculated membership values correspond to the firing strength of 24 "IF–THEN" rules in the hidden layer. Finally, the output layer defuzzifies the fuzzy values as the crisp value to be used in practice.

The input data of T1FNN are chosen as the distances and the angles between the robot and goal position (or nearest obstacle) as follows:

- d_g is the distance between the robot and the goal position.
- d_o is the distance between the robot and the nearest obstacle.
- $_g$ is the angle between the robot orientation and the direction to the goal position.
- $_o$ is the angle between the robot orientation and the direction to the nearest obstacle.

Remark 1: The input data of T1FNN can be chosen in an appropriate way depending on the task such as shooting behavior or posture stabilization. In particular, suppose that we choose the goal angle of input data as $\theta_g = 2\varphi - \theta_{\text{goal}} - \theta$, where $\varphi = \tan^{-1}(dy_g/dx_g)$; dy_g and dx_g are distances between the robot and goal position along the Y –axis and X-axis, respectively; and θ_{goal} is the desired orientation of robot when the robot arrives at the goal position. Then, the *posture stabilization* instead of *position stabilization* can be achieved.

Remark 2: The objective of the proposed method is to achieve both obstacle avoidance and position stabilization for the wheeled mobile robot. To this end, additional information on the angles between the robot orientation and the obstacles other than those between the robot orientation and the nearest obstacle is not necessary. In addition, the advantage of the proposed method over the previous methods with hierarchical structure lies in the fact that the proposed system structure can be much simplified. Therefore, we have chosen four input data and two output data as in Fig. 2, considering the implementation of the overall system.

Once the input data are determined, the membership values can be calculated by using the membership functions. The membership values imply the grade of linguistic terms in the system. For example, the membership value $\mu_{Ai}(d_g)$ represents the grade of the distance between the robot and the goal position; if that distance is far, then $\mu_{A1}(d_g)$ becomes large and $\mu_{A2}(d_g)$ becomes small and the summation of these two [10]membership values is always one. The same can be said to the other membership values such as $\mu_{Bj}(d_o)$, $\mu_{Ck}(\theta_g)$ and $\mu_{Dl}(\theta_o)$. The membership functions to calculate the membership values are expressed as follows:

$$\begin{cases} \mu_{A_i}(d_g) & = \dfrac{1}{1+e^{-a_{A_i}(d_g - c_{A_i})}} \\ \mu_{B_j}(d_o) & = \dfrac{1}{1+e^{-a_{B_j}(d_o - c_{B_j})}} \\ \mu_{C_k}(\theta_g) & = \dfrac{1}{1+e^{-a_{C_k}(\theta_g - c_{C_k})}} \end{cases} \tag{8}$$

Where subscripts A_i, B_j,andC_k are the afore mentioned linguistic terms for $i, j, k = 1{,}2$ and a_{Ai}, a_{Bj}, a_{Ck},c_{Ai},c_{Bj},c_{Ck} are the design parameters that are learned by the genetic algorithm. The membership function $\mu_{Dl}(\theta_o)$ is defined by the following Gaussian functions depending on whether the situation between the robot and the nearest obstacle is "left" or "right":

1)If θ_o is "left," then

$$\begin{cases} \mu_{D_1}(\theta_o) = e^{\frac{-\theta_o^2}{2\sigma_{d_1}^2}} \\ \mu_{D_2}(\theta_o) = 0 \\ \mu_{D_3}(\theta_o) = 1 - (\mu_{D_1}(\theta_o) + \mu_{D_2}(\theta_o)). \end{cases} \tag{9}$$

2)If θ_o is "right," then

$$\begin{cases} \mu_{D_1}(\theta_o) & = 0 \\ \mu_{D_2}(\theta_o) & = e^{\frac{-\theta_o^2}{2\sigma_{d_2}^2}} \\ \mu_{D_3}(\theta_o) & = 1 \quad (\mu_{D_1}(\theta_o) + \mu_{D_2}(\theta_o)) \end{cases} \tag{10}$$

Here, σ_{d1} and σ_{d2} are the design parameters that determine the width of the membership function. When θ_o is neither "left" nor "right" (i.e., the nearest obstacle is placed right in front of the mobile robot), then either $\mu_{D1}(\theta_o)$ or $\mu_{D2}(\theta_o)$ can be chosen to become 1 and the remaining membership values become zero. Here, the membership value $\mu_{D3}(\theta_o)$ has the property that it is inversely proportional to the degree of the influence which the nearest obstacle has on the mobile robot.

The calculated membership values correspond to the firing strength of *jilt* rule in the hidden layer. As shown in Fig. 3, since the input data d_g, d_o and θ_g have two membership values and θ_o has three membership values, the hidden layer of T1FNN consists of 24 Takagi–Surgeon-type fuzzy "IF–THEN" rules. The *jilt* rule can be represented as follows:

Rule: If d_g is A_i,$d_o$is B_j, θ_g is C_k and θ_o is D_l then $fv_{ijkl} = Vijkl$ and $fw_{ijkl} =$ Ù*ijkl*

where $i,j,k = 1{,}2$; $l = 1{,}2{,}3$; A_i,B_j,C_k and D_l are fuzzy sets; and V_{ijkl} and Ù$_{ijkl}$ are constants. On the basis of rules, *ijkl*th strength is calculated as

$$WijklT1FNN = \mu Ai\ (dg)\mu Bj(do)\mu Ck\ (\theta g)\mu Dl\ (\theta o). \quad (11)$$

The obtained output plays the role of the control input of the mobile robot, which can make the robot move from the initial position to the goal position without collision.

The T1FNN has many design parameters such as a, c, σ, $f_{vij\,k\,l}$ and $f_{wij\,k\,l}$. The genetic algorithm is used to tune the design parameters. The genetic algorithm can find optimal solution in a highly nonlinear and complex space and is also a parallel and global search technique. For the learning of T1FNN parameters, each T1FNN parameter is encoded by binary values. The real values of the parameters are limited to the range (-5, 5). The candidate solutions are represented by binary strings as follows:

$$abA1, abA2, cbA1, cbA2, abB1, aBb2,\ cBb\ 1, cBb\ 2, aCb1, aCb\ 2, cCb\ 1, cbC2 \quad (12)$$

$$\sigma, fv1111, \ldots, fv2223, fw1111, \ldots, fw2223 \quad (13)$$

where the superscript "b" stands for the binary value. In addition, the fitness function is defined as

$$F = \frac{10}{\gamma + 6d^2 + d_1 + d_2 + 2v_{\text{fit}} + 0.5w_{\text{fit}} + \delta + 60\rho}. \quad (14)$$

Here, γ is a small positive constant; d, d_1 and d_2 are the average distances from the robot to the target and v_{fit} and w_{fit} are linear and angular velocities of robot, respectively; δ is function of distance between the robot and obstacle; and ρ is a Boolean variable, where $\rho = 0$ if the robot has achieved velocities and $\rho = 1$ otherwise.

By using T1FNN, we can consider the various situations between the robots and the obstacles via the fuzzy method and we can unify the numerical input and output data as a single structure system via the neural network such that the desired linear and angular velocities can be obtained. However, the performance of T1FNN has the following limitations. First, as T1FNN uses the crisp set as the membership values, the output of T1FNN is decided based on the situation between the robots and the environment, which implies that T1FNN canb ejust considered as a simple function of the input and output data. Thus, T1FNN cannot reduce the influence of the uncertainties effectively and accordingly, the performance of T1FNN is degraded in the face of the uncertainties or unstructured environments such as the obstacles. Second, the angular velocity of T1FNN is increased during the obstacle avoidance and can become even saturated, which implies that more control effort is required and it may cause the system to become unstable. Actually, the large oscillating behaviors can be seen in both simulation and experimental results of T1FNN. Therefore, we need to design the proposed method and then, we can expect improved performance such as the robustness against the uncertainties and the less control effort.

Proposed System: The membership values in the input layer of proposed system in the form of fuzzy set can reduce the influence of the uncertainties, we can expect improved performance by employing proposed system. In the case of proposed system, the inference engine uses the aforementioned fuzzy input set as the membership values. Therefore, in order to reduce the influence of uncertainties, the input data are fuzzified and the membership values are calculated by the membership functions in the input layer of proposed system. Since the membership functions of proposed system evaluate the fuzzy set as output unlike the membership functions of T1FNN, proposed system has three kinds of memberships, i.e., "normal," "lower," and "upper." These three membership functions produce three different values as output and thus, the proposed system can have fuzzy set as membership values unlike T1FNN. Each membership function consists of "normal", "lower" and" upper" membership functions. Whereas "normal"membershipfunctionisthesameasthatoftheT1FNN, "lower" membership function $\mu(\bullet)$ and "upper" membership function $\mu(\bullet)$ are defi$\underline{\text{n}}$ed through the introduction of additional parameters such as *æ* and *æ*, where *æ* means th$\underline{\text{e}}$ shape parameter of "lower" membership functions and *æ* means that of "upper" membership functions.

The "upper" and "lower" membership functions of proposed system are introduced for d_g, d_o, θ_g as follows:

$$\begin{cases} \overline{\mu}_{A_i}(d_g) = \dfrac{1}{1 + e^{-a_{A_i}(d_g + \overline{\zeta}_{d_g} - c_{A_i})}} \\ \underline{\mu}_{A_i}(d_g) = \dfrac{1}{1 + e^{-a_{A_i}(d_g + \underline{\zeta}_{d_g} - c_{A_i})}} \end{cases} \quad (15)$$

$$\begin{cases} \overline{\mu}_{B_j}(d_o) = \dfrac{1}{1 + e^{-a_{B_j}(d_o + \overline{\zeta}_{d_o} - c_{B_j})}} \\ \underline{\mu}_{B_j}(d_o) = \dfrac{1}{1 + e^{-a_{B_j}(d_o + \underline{\zeta}_{d_o} - c_{B_j})}} \end{cases} \quad (16)$$

$$\begin{cases} \overline{\mu}_{C_k}(\theta_g) = \dfrac{1}{1 + e^{-a_{C_k}(\theta_g + \overline{\zeta}_{\theta_g} - c_{C_k})}} \\ \underline{\mu}_{C_k}(\theta_g) = \dfrac{1}{1 + e^{-a_{C_k}(\theta_g + \underline{\zeta}_{\theta_g} - c_{C_k})}} \end{cases} \quad (17)$$

The additional parameters *æ bar*and *æ* are included in these equations, which play the role of the uncertain means. In the case ofθ_o, the "lower" and "upper" membership functions are defined as follows.

1)If θ_o is "left," then

$$\begin{cases} \overline{\mu}_{D_1}(\theta_o) = e^{\frac{-\theta_o^2}{2\sigma_{d_1}^2 + \overline{\zeta}_{\theta_o}}}, \underline{\mu}_{D_1}(\theta_o) = e^{\frac{-\theta_o^2}{2\sigma_{d_1}^2 + \underline{\zeta}_{\theta_o}}} \\ \overline{\mu}_{D_2}(\theta_o) = 0, \underline{\mu}_{D_2}(\theta_o) = 0 \\ \overline{\mu}_{D_3}(\theta_o) = 1 - (\overline{\mu}_{D_1}(\theta_o) + \overline{\mu}_{D_2}(\theta_o)) \\ \underline{\mu}_{D_3}(\theta_o) = 1 - (\underline{\mu}_{D_1}(\theta_o) + \underline{\mu}_{D_2}(\theta_o)). \end{cases} \tag{18}$$

2)If θ_o is "right," then

$$\begin{cases} \overline{\mu}_{D_1}(\theta_o) = 0,\ \underline{\mu}_{D_1}(\theta_o) = 0 \\ \overline{\mu}_{D_2}(\theta_o) = e^{\frac{-\theta_o^2}{2\sigma_{d_2}^2 + \overline{\zeta}_{\theta_o}}},\ \underline{\mu}_{D_2}(\theta_o) = e^{\frac{-\theta_o^2}{2\sigma_{d_2}^2 + \underline{\zeta}_{\theta_o}}} \\ \overline{\mu}_{D_3}(\theta_o) = 1 - (\overline{\mu}_{D_1}(\theta_o) + \overline{\mu}_{D_2}(\theta_o)) \\ \underline{\mu}_{D_3}(\theta_o) = 1 - (\underline{\mu}_{D_1}(\theta_o) + \underline{\mu}_{D_2}(\theta_o)). \end{cases} \tag{19}$$

The shape of μ_D (θ_o) becomes different depending on the value of θ_o unlike the other membership functions. In particular, $\mu_D(\theta_o)$ is defined by the Gaussian functions, which can be adjusted by the additional parameters

Experimental Results: The proposed system has undergone various experiments to validate the proposed method. The experiments are carried out using the FNN designed in python script. The experimental setup has been developed as shown in Fig. 4, which receives the information of the robot posture and the obstacles position from the RFID Reader and an ultrasonic sensor. The situation between the robot and the obstacles is observed and then, the necessary data for the verification of the performance of the proposed method are saved for the intelligence of the wheeled robot. In the python program, the desired linear and angular velocities of the robot are calculated by using this proposed system and the calculated linear and angular velocities are sent to the robot by wireless communication system.

The wheel velocities of the robot are controlled by the Motor controller (L293D) in order to achieve position stabilization and obstacle avoidance. The robot platform is implemented by using a Raspberry Pi as the CPU.

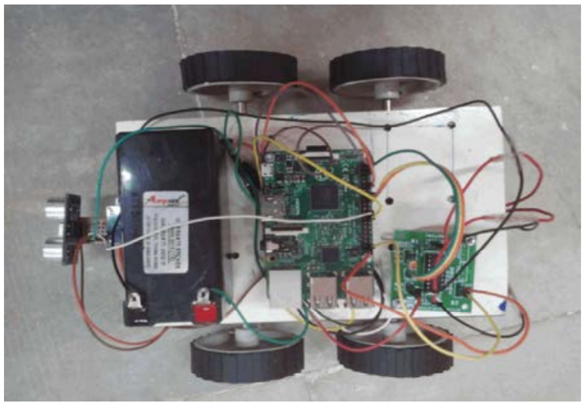
Fig. 4: Hardware developed for experiment

The experimental result tells that the angular velocity of proposed system is less than that of T1FNN. In addition, the angular velocity of T1FNN is saturated, which is not so in the case of Proposed system.

CONCLUSION

An obstacle avoidance method in the position stabilization for the wheeled mobile robot system has been proposed using proposed system. In the proposed system method, we redefined the membership functions through the addition of the uncertain means and standard deviation, which can have the form of fuzzy set membership values. As shown in experimental results, the robot using proposed system can achieve both obstacle avoidance and position stabilization in a satisfactory way. The comparisons of the final distance errors and the moved distances show that the performance of the robot using proposed system is better than that of the previous T1FNN. That is, the proposed system can guarantee the smaller final distance errors and the shorter moved distances than the previous T1FNN. This implies that the robot can follow the shorter path and can achieve smoother movement during obstacle avoidance. In addition, it requires less control effort for the task. Therefore, we can conclude that the proposed method the wheeled mobile robot is a more practical method and works better than the previous T1FNN, as verified in experimental result. These advantages could be obtained due to the novel fuzzy neural network structure for the wheeled mobile system, which should achieve several tasks at the same time. Thus, it can be expected that more studies involving proposed system for mobile robot systems will take place in the future.

REFERENCES

1. Chwa, D., 2004. Sliding mode tracking control of nonholonomic wheeled mobile robots in polar coordinates, IEEE Trans. Control Syst. Technol., 12(4): 637–644.
2. Chwa, D., 2010. Tracking control of differential-drive wheeled mobile robots using backstepping -like feedback linearization, IEEE Trans. Syst., Man, Cybern. A, Syst., Humans, 40(6): 1285-1295.
3. Chwa, D., 2012. Fuzzy adaptive tracking control of wheeled mobile robots with state-dependent kinematic and dynamic disturbances, IEEE Trans. Fuzzy Syst., 20(3): 587-593.
4. Baturone, I., F.J. Moreno-Velo, V. Blance and J. Ferruz, 2008. Design of embedded DSP-based fuzzy controllers for autonomous mobile robots, IEEE Trans. Ind. Electron., 55(2): 928-936.
5. Sanchez-Solano, S., A.J. Cabrera, I. Baturone, F.J. Moreno-Velo and M. Brox, 2007. FPGA implementation of embedded fuzzy controllers for robotic applications, IEEE Trans. Ind. Electron., 54(4): 1937-1945.
6. Rusu, P., E.M. Petriu, T.E. Whalen, A. Cornell and H.J.W. Spoelder, 2003. Behavior-based neuro-fuzzy controller for mobile robot navigation, IEEE Trans. Instrum. Meas., 52(4): 1335-1340.
7. Chatterjee, A., K. Pulasinghe, K. Watanabe and K. Izumi, 2005. A particleswarm-optimized fuzzy-neural network for voice-controlled robot systems, IEEE Trans. Ind. Electron., 52(6): 1478-1489.
8. Zhu, A. and S.X. Yang, 2007. Neurofuzzy-based approach to mobile robot navigation in unknown environments, IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., 37(4): 610-621.
9. Godjevac, J. and N. Steele, 1999. Neuro-fuzzy control of a mobile robot, Neurocomputing, 28(1): 127-143.
10. Er, M.J. and C. Deng, 2004. Online tuning of fuzzy inference systems using dynamic fuzzy q-learning, IEEE Trans. Syst., Man, Cybern. B, Cybern., 34(3): 1478-1489.