Position Control of Robotino Mobile Robot Using Fuzzy Logic

S. E. Oltean[1], M. Dulău[1], R. Puskas[2]
[1]"Petru Maior" University of Tg. Mures, soltean@engineering.upm.ro, mdulau@engineering.upm.ro
[2]S.C. Marquardt Schaltsysteme S.C.S. Sibiu, puskas_roberst@yahoo.com

***Abstract*-The appearance of fuzzy logic theory led to new trends in robotics field and solves various typical problems. In fact, fuzzy logic is a good alternative for increasing the capabilities of autonomous mobile robots in an unknown dynamic environment by integrating human experience. The paper presents the design and implementation of the position control using fuzzy logic for an omni directional mobile robot. The position control of the mobile robot was tested remotely using the Festo Robotino platform and Matlab environment. The obstacle avoidance and mobile robot odometry were also studied in this work.**

I. Introduction

Since the first industrial robot began its work at the Ford factories in 1959, research in robot technology has been performed intensively.

At the beginning the robots replaced some employees involved in jobs with repetitive tasks laying the foundation area of industrial robots. Later on, the tele-operation permuted the human control of the robots and the diversification of the tasks.

Now, due to the technological progress the robotics field covers not only the industrial applications but also a wide range of others purposes. Thus, advanced robots get increasingly autonomous skills to be able to participate in every human life.

Autonomous robots are used in various areas like Industry, Service and care, Agriculture, and Hospitals, where the advanced robots are dealing with a dynamic human environment, which require human interaction and reaction to signals, namely a high degree of automation. By comparison, industrial robots are used in static and known environment.

The high degree of automation and autonomy of robots is obtained by the intersection of research from various fields such as social sciences, artificial intelligence, robotics, automation, computer science [1].

A new step in the field of robotics dealing with uncertainties and human environment was the introduction of fuzzy logic. Nowadays of fuzzy logic is used in various fields, but in case of mobile robots it solves some typical problems [2,3]:

- autonomous navigation;
- behavioral control;
- speed control;
- map creation in unknown environments;
- tracking of natural features in robot's environment;
- vision-based navigation.

Of course, the intersection between the field of fuzzy logic and mobile robotics is becoming more evident. Furthermore, most reported designs rely on intelligent control approaches such as fuzzy logic control and neural networks [5, 6].

In the autonomous mobile robots tasks the accurate localization is a key prerequisite for successful navigation in large scale environments, particularly when global models are used, such as maps, CAD models, drawings, and topological descriptions.

For the self-localization of the mobile robot a large range sensor with superior performance is needed, eg scanning laser range finder developed for robotics applications. For the experimental stage we had the Festo Robotino Platform without such a sensor. So, we designed a fuzzy logic position control of a mobile robot using just the optical shaft encoders and infrared distance sensors for the measurements.

After implementation of the algorithm in the Matlab language, our robot will be able to perform positioning tasks in a dynamic human environment even in the presence of obstacles. For the implementation of the algorithm we used also the Robotino Matlab Toolbox.

Fig. 1 shows the test environment, in which the robot is to be functioning. The environment consists of a single room with known geometric dimensions.

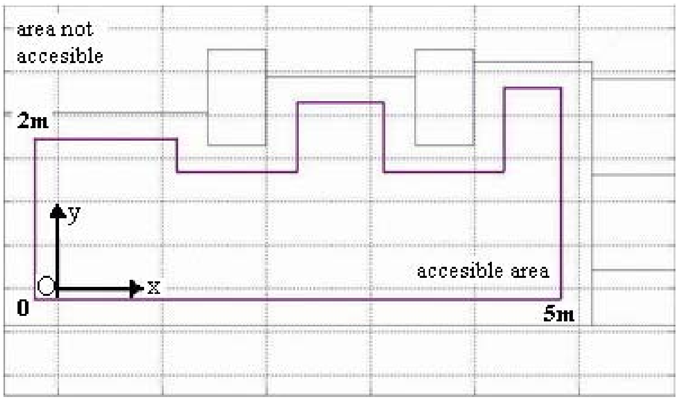

Figure 1. Illustration of the test environment.

In the next sections of the paper we present a short description of the Festo Robotino mobile robot with some technical details, the positioning algorithm and block diagram based on fuzzy logic control, experimental results and finally some conclusions and further development possibilities.

II. Festo Robotino Platform

Festo Robotino is a holonomic mobile robot with three omni directional drive units, meaning that the controllable degrees of freedom equal the total degrees of freedom of the robot. Powered by DC motors equipped with optical shaft encoders the Robotino can reach speeds of up to 10km/h. Furthermore, interchangeable pinions allow for custom gearing from 1:4 to 1:16 [7].

A. Technical specifications

Some of the Robotino technical specifications are listed in Table 1 [7].

TABLE I
TECHNICAL SPECIFICATION OF THE FESTO ROBOTINO HARDWARE PLATFORM

No.	Robotino
1	Robot dimensions: - 370mm diameter, 210 height, 11kg overall weight and 5kg maximum payload.
2	Sensors: - nine Sharp GD2D120 infrared distance sensors, analogue inductive sensor, bumper with integrated sensor, two optical sensors, webcam with USB interface, three optical shaft encoders
3	Embedded Controller: - PC104 MOPSlcdVE processor of 300MHz, SDRAM 64MB, 128MB Compact flash card with C++ libraries for accessing the Robotino, Wireless LAN interface board. Interfaces: Ethernet, 2 x USB, 2 x RS-232, Keyboard and mouse, parallel port.
4	I/O interface card: - outputs for controlling the three omni directional drive units, 10 analogous inputs (0 − 10V, 50 Hz), two analogous outputs, 16 digital inputs/outputs, three relays

Fig. 2 shows the robot and the robot depicted to highlight theinfrared sensors and DC motors.

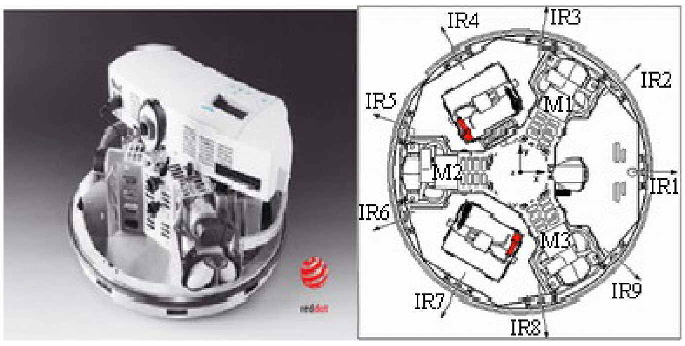

Figure 2. Festo Robotino with infrared sensors and DC motors.

The operating system of the Robotino is divided into two layers, being a regular Linux layer and a Real Time Linux (RTLinux) layer. All hardware access goes through the RTLinux layer, while the Linux layer provides standard user space. Also, the real time control of the robot can be made remotely from another PC using the RTLinux layer and wireless communication.

Bundled with Robotino platform, Festo provides an Application Programming Interface (API) called RobotinoCom. The API offers the functions for wireless communication and for accessing sensors and actuators [7].

In our case, Robotino control procedure from Matlab language via wireless communication is simplified by using the Robotino Matlab Toolbox. The information about the Robotino position is obtained using the optical shaft encoders and dead reckoning method (robot odometry), while the obstacles avoidance require the infrared distance sensors. These infrared sensors allow the obstacles detection located at distances less than 30cm.

B. Robot odometry

Robot odometry means the use of data from the movement of the three actuators to estimate the robot position over time. In the current design problem we will use the odometry to estimate the mobile robot position relative to the known starting location.

So, using the kinematical model of the robot and by integrating the robot velocity we find the mobile robot position. Optical encoders will provide the basic data to estimating the robot's position and orientation towards starting location. This is a cheap and quick solution with good accuracy to short-term if we consider the assumptions:

- the friction between the wheels and the floor is infinite;
- the centre of mass is located at the geometrical centre of the robot;
- the distance between the centre of the robot and the centre of the wheel currently touching the ground is constant.

Fig. 3 shows the kinematical model of the robot.

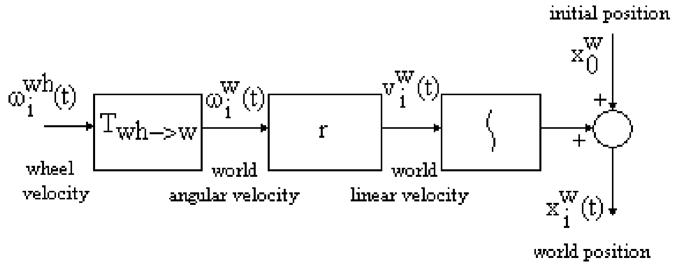

Figure 3. Kinematical model of the mobile robot.

Theory of geometry and rotation is used to develop the models from Fig. 3. Referring to the geometry used for deriving the Robotino kinematical model we defined two coordinate systems. First coordinate system is fixed to the environment, called world coordinate system. The second system is fixed to the mobile robot and it is named robot coordinate system.

To be able to distinguish between velocities relative to the two coordinate systems mentioned above the following definitions need to be introduced:

- robot velocity is the velocity relative to the robot coordinate system;
- world velocity is the velocity relative to the world coordinate system.

The relationship between wheels velocities and world velocities can now be derived. We made the calculus in two stages. First stage consists in the transformation (1) from the wheels velocities to the robot velocities or inverse. We defined the following variables:

- ω_i is the angular velocity of the wheel i (the angular velocities of the wheels describe the vector ω^{wh});
- v_i is the velocity of each individual wheel (the translational velocities of the wheels describe the vector v^{wh});

- r the radius of the wheel;
- R the distance from the wheel to the centre of the robot;
- x_r and y_r characterize the robot coordinate system;
- Ω the tangential rotational velocity of the robot;
- φ_i describes the rotation of the wheel shaft compared to the robot coordinate system.

$$\omega^{wh} = \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix} = \frac{1}{r} \cdot \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} = \frac{1}{r} \cdot \begin{bmatrix} -\sin(\varphi_1) & \cos(\varphi_1) & R \\ -\sin(\varphi_2) & \cos(\varphi_2) & R \\ -\sin(\varphi_3) & \cos(\varphi_3) & R \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_r(t) \\ \dot{y}_r(t) \\ \Omega(t) \end{bmatrix} \quad (1)$$

Determining robot velocities from wheels velocities can be done by inverting the matrix.

The second stage treats the relationship between the wheels velocities and the velocity of the robot expressed in the world coordinate (2).

$$\omega^{wh} = \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix} = \frac{1}{r} \cdot \begin{bmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{bmatrix} = \frac{1}{r} \cdot v^{wh}$$

$$\omega^{wh} = \frac{1}{r} \cdot \begin{bmatrix} -\sin(\theta+\varphi_1) & \cos(\theta+\varphi_1) & R \\ -\sin(\theta+\varphi_2) & \cos(\theta+\varphi_2) & R \\ -\sin(\theta+\varphi_3) & \cos(\theta+\varphi_3) & R \end{bmatrix} \cdot \begin{bmatrix} \dot{x}_w(t) \\ \dot{y}_w(t) \\ \Omega(t) \end{bmatrix} \quad (2)$$

where

- x_w and y_w characterize the world coordinate system;
- θ is the rotation angle of the robot coordinate system compared to the world coordinate system.

In the next sections we simplified the formulas by replacing the transformation matrix from the wheels velocities to the world angular velocity with $T_{wh \to w}$.

Using the expressions derived above, it is possible to determine the robot's world velocity from the measurements of the angular velocities of the three wheels by inverting the matrix $T_{wh \to w}$.

$$\begin{bmatrix} \dot{x}_w(t) \\ \dot{y}_w(t) \\ \Omega(t) \end{bmatrix} = r \cdot \left(T_{wh \to w}\right)^{-1} \cdot \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix} \quad (3)$$

Equations (1), (2) and (3) are some basic relations to find the robot position in the human environment. Dead reckoning method give us a simple mathematical procedure to estimating the current position of the robot based on previously determined position when the velocity and elapsed time until the current position are known.

The estimated position by moving the robot from the time moment t_0 at the time moment t_1 is determined using (4).

$$\begin{bmatrix} x_w(t_1) \\ y_w(t_1) \\ \theta(t_1) \end{bmatrix} = \begin{bmatrix} x_{w,0} \\ y_{w,0} \\ \theta_0 \end{bmatrix} + r \cdot \int_{t0}^{t1} \left(\left[T_{wh->w}\right]^{-1} \cdot \begin{bmatrix} \omega_1(t) \\ \omega_2(t) \\ \omega_3(t) \end{bmatrix} dt \right) \quad (4)$$

where

- $x_w(t_1)$ and $y_w(t_1)$ are the world coordinate system at time moment t_1;
- $\theta(t_1)$ the orientation angle at moment t_1;
- $x_{w,0}$ and $y_{w,0}$ are the world coordinate system at moment t_0;
- θ_0 the orientation angle at moment t_0.

A disadvantage of dead reckoning is that since new positions are calculated solely from previous positions, the errors of the process are cumulative, so the errors in the position estimation grow with time.

III. The Position Control System For The Robotino Mobile Robot

The block diagram of the position control system for the Robotino mobile robot is shown in Fig. 4. The control scheme has two feedback loops. The main feedback loop is important for the robot position control and provides the information about the current position relative to the desired position, while the other loop uses the infrared distance sensors to detect the obstacles in the human environment and to modify the optimal trajectory so that the obstacle to be avoided.

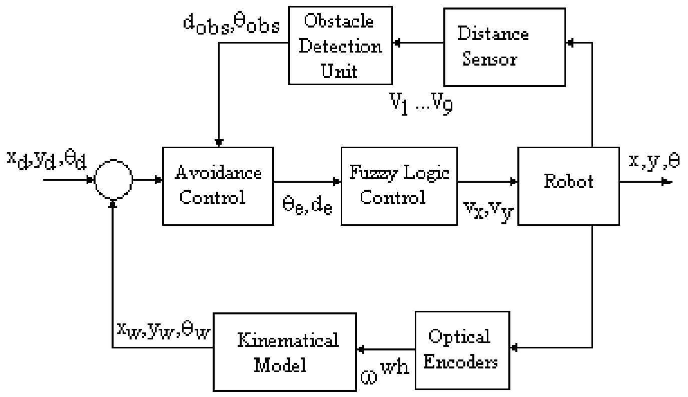

Figure 4. The block diagram of the position control system.

The new variables which appear in the Fig. 4 are the desired position and orientation of the robot x_d, y_d and θ_d, the orientation angle error θ_e, distance error d_e, distance to the obstacle d_{obs}, angle of the obstacle detection θ_{obs}, voltages from the distance sensors $V_1 \ldots V_9$, linear velocities v_x and v_y of the robot in the Ox and Oy directions, position and orientation of the robot x, y, θ.

The robot module contains the omni drive component which transforms the robot linear velocities in the wheels velocities, the internal PID controller for the wheels velocities, the Dunker DC motors which drive the robot, the mechanics and other additional components. The block diagram highlights also the robot sensors, infrared distance sensors and optical shaft encoders.

The kinematical model module estimates the current position of the robot using the dead reckoning method using (4). This module has as inputs the wheels angular velocities determined using the optical shaft encoders and as outputs values which characterize the current bi-dimensional position and the robot orientation.

The obstacle detection unit uses nine distance sensors to detect obstacles within a distance up to 30cm. Because the sensors are placed at an angle of 40 degrees, there are so called invisible points. Thus, the detection was tested with large objects.

Depending on the voltages measured from the distance sensors (0.4V-2.54V values represent 4cm-30cm distances) the detection module provides the distance d_{obs} and angle θ_{obs} at which the obstacle is (if any obstacle exists).

Avoidance control module computes the orientation angle error θ_e and distance error d_e knowing the desired position of the robot and the current position using (5).

$$\begin{aligned} d_e &= \sqrt{(x_d - x_w)^2 + (y_d - y_w)^2} \\ \theta_e &= \theta_w - \theta_d \end{aligned} \tag{5}$$

If any obstacle is detected by the detection unit then to the orientation angle θ_e error is added the obstacle angle θ_{obs}.

The fuzzy logic controller has the role to lead the robot to the desired position. The design of fuzzy controller can be resumed to choosing and processing the inputs and outputs of the controller and designing its four component elements (the rule base, the inference engine, the fuzzification and the defuzzification interface) [8, 9].

We choose as inputs into the controller the orientation angle error θ_e and distance error d_e. Also we consider two outputs: the velocities of the robot in the world coordinate system which are the linear velocities v_x and v_y in the Ox and Oy directions.

The universe of discourse of the variables (that is, their domain) was considered to cover the practical values. We used 13 membership functions for the orientation angle error and for the outputs and 7 membership functions for the distance error. All these functions have a triangular shape.

The linguistic values that characterize the linguistic variable are:

- for the orientation angle error: Negative Infinite NIN, Negative Very Large NVL, Negative Large NL, Negative Middle NM, Negative Small NS, Negative Very Small NVS, Zero ZE, Positive Very Small PVS, Positive Small PS, Positive Middle PM, Positive Large PL, Positive Very Large PVL, Positive Infinite PIN;
- for the distance error: Zero Z, Very Close VCL, Close CL, Middle M, Far F, Very Far VF, Infinite IN;
- for the linear velocities: Turbo Backward TBW, Very Fast Backward VFSBW, Fast Backward FSBW, Backward BW, Slow Backward SLBW, Very Slow Backward VSLBW, Stop S, Very Slow Forward VSLFW, Slow Forward SLFW, Forward FW, Fast Forward FSFW, Very Fast Forward VFSFW, Turbo Forward TFW.

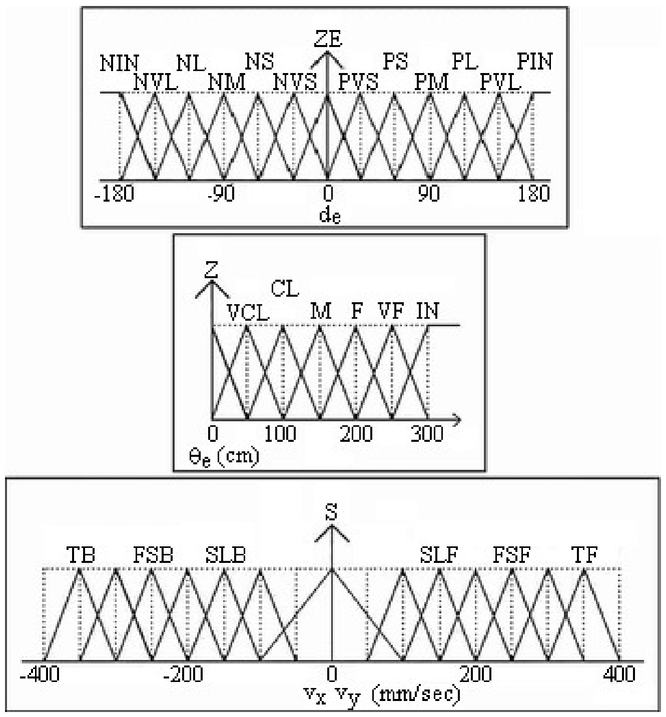

Figure 5. Membership functions for the fuzzy controller.

The fuzzy controller implements a rule base made of a set of IF-THEN type rules. These rules were determined heuristically based on the knowledge of the plant.

The resulting rule tables for the v_x and v_y linguistic variables are shown in Fig. 6.

$\theta_e \backslash d_e$	Z	VCL	CL	M	F	VF	IN
NIN	S	VSLBW	SLBW	BW	FSBW	VFSBW	TBW
NVL	S	BW	BW	FSBW	FSBW	VFSBW	TBW
NL	S	VSLBW	SLBW	BW	BW	FSBW	VFSBW
NM	S	S	S	S	S	S	S
NS	S	VSLFW	SLFW	FW	FW	FSFW	VFSFW
NVS	S	FW	FW	FSFW	FSFW	VFSFW	TFW
ZE	S	VSLFW	SLFW	FW	FSFW	VFSFW	TFW
PVS	S	FW	FW	FSFW	FSFW	VFSFW	TFW
PS	S	VSLFW	SLFW	FW	FW	FSFW	VFSFW
PM	S	S	S	S	S	S	S
PL	S	VSLBW	SLBW	BW	BW	FSBW	VFSBW
PVL	S	BW	BW	FSBW	FSBW	VFSBW	TBW
PIN	S	VSLBW	SLBW	BW	FSBW	VFSBW	TBW

$\theta_e \backslash d_e$	Z	VCL	CL	M	F	VF	IN
NIN	S	S	S	S	S	S	S
NVL	S	VSLFW	VSLFW	SLFW	SLFW	FW	FSFW
NL	S	SLFW	FW	FSFW	FSFW	VFSFW	TFW
NM	S	VSLFW	SLFW	FW	FSFW	VFSFW	TFW
NS	S	SLFW	FW	FSFW	FSFW	VFSFW	TFW
NVS	S	VSLFW	VSLFW	SLFW	SLFW	FW	FSFW
ZE	S	S	S	S	S	S	S
PVS	S	VSLBW	VSLBW	SLBW	SLBW	BW	FSBW
PS	S	SLBW	BW	FSBW	FSBW	VFSBW	TBW
PM	S	VSLBW	SLBW	BW	FSBW	VFSBW	TBW
PL	S	SLBW	BW	FSBW	FSBW	VFSBW	TBW
PVL	S	VSLBW	VSLBW	SLBW	SLBW	BW	FSBW
PIN	S	S	S	S	S	S	S

Figure 6. Rule tables for the fuzzy controller.

Some examples of IF-THEN rules for the linear velocities in the Ox and Oy directions are the following:

IF θ_e is negative small NS and d_e is close CL THEN v_x is slow forward SLFW

IF θ_e is negative small NS and d_e is close CL THEN v_y is forward FW

The rules quantifies the situation where the mobile robot is close to the desired position with a negative small orientation angle error, hence a slow forward velocity in the Ox direction and a little more forward velocity in the Oy direction are needed to compensate the orientation angle error and to advance towards the desired position.

The min-max inference engine was chosen, whish for the premises, uses maximum for the OR operator and minimum for the AND operator. The conclusion of each rule, introduced by THEN, is also done by minimum. The final conclusion for the active rules is obtained by the maximum of the considered fuzzy sets [8].

To obtain the crisp outputs, the centre of gravity defuzzification method is used. These crisp values are the resulting controller outputs.

IV. Experimental Results

The position control system was tested on the Robotino mobile robot with various scenarios: robot moving forward with/without obstacles, robot moving backward with/without obstacles, position control with one or more obstacles, position control in the human environment with static or dynamic obstacles.

For example in the following paragraphs are presented the experimental results of positioning the mobile robot in human environment with two static obstacles. Top view of the Robotino moving forward from the start point to the goal and avoiding the obstacles are shown in Fig. 7.

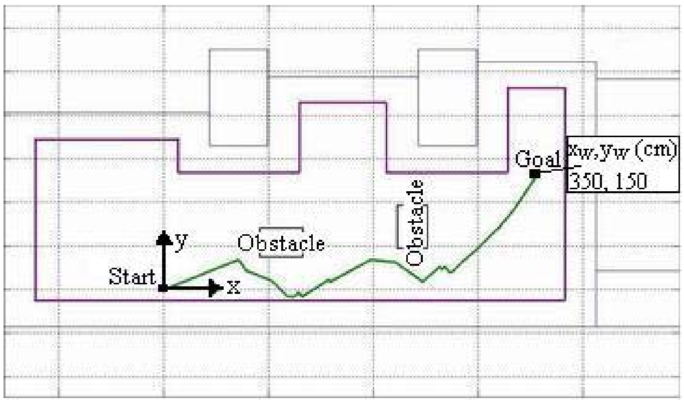

Figure 7. Top view of the Robotino plane trajectory.

The distance d_{obs} and the angle θ_{obs} at which the obstacle is were calculated using the voltage measured in time from nine infrared distance sensors.

The functional signals for the fuzzy logic controller meaning the orientation angle error θ_e, the distance error d_e and the linear velocities v_x and v_y (representing the outputs from the controller) are also presented in Fig. 8.

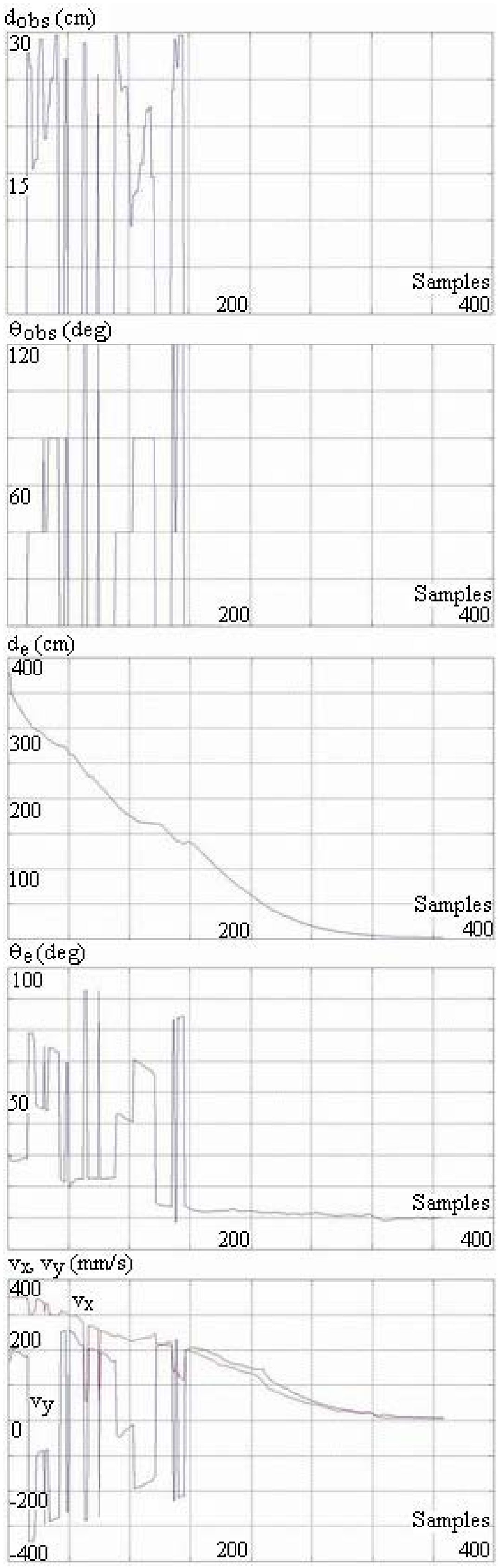

Figure 8. The functional signals of the controller.

The angular wheels velocities, the Dunker DC motors currents provided by the internal PID controllers and the position of the robot in the world coordinate system are shown in Fig. 9.

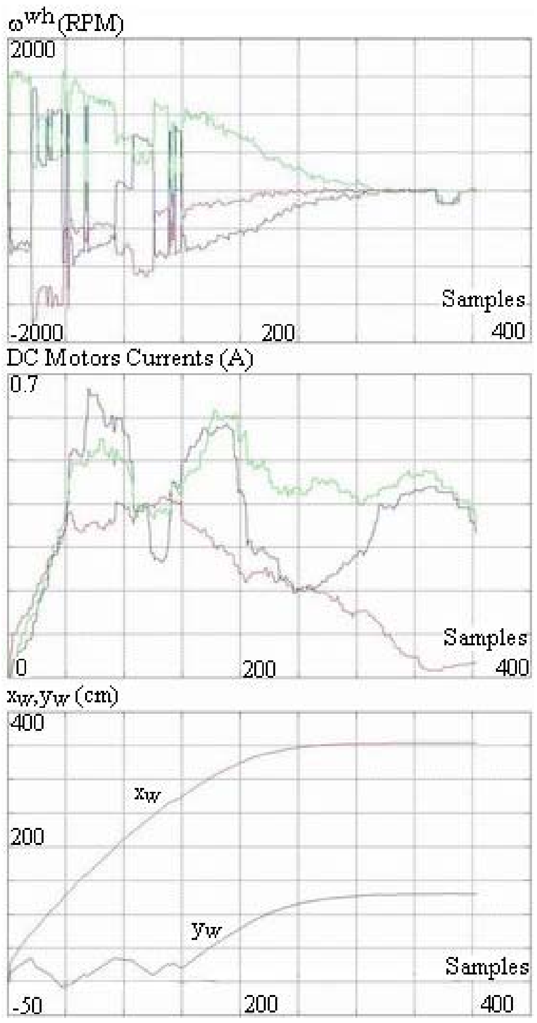

Figure 9. The angular wheels velocities and world coordinate system.

V. Conclusions

This paper studied the implementation of the position control for a mobile robot based on fuzzy logic. The experiments were made using the Festo Robotino platform. We treated also the obstacles avoidance in the human environment.

The fuzzy controller allowed the use of heuristics (which model the way a human would control the process) via the use of rule table. Since we generally know the way to control the robot, the heuristics we chose in the design of the fuzzy controller proved very useful. The fuzzy controller has as inputs the orientation angle error and the distance error, and as outputs the linear velocities of the robot in the world coordinate system which are transformed by the omni drive module to the wheels velocities.

Movement status information is obtained through robot sensors. Optical shaft encoders and dead reckoning algorithm are used for current position estimation, while infrared distance sensors are useful for the obstacle detection.

The experimental results confirm that the controller can achieve our objective with some limitation. These operating limitations are given by using the encoders to estimate the robot position. The straightforward way to track the robot's movements by simply integrating velocity data (known as odometry) may become a source of error (systematic or non-systematic).

However, several extensions can be made to the control structure, such as to increase the tracking accuracy and the performance level. One improvement to the system is to find a method in which the odometry data is supplemented by other sensory information (eg sonar, laser scanning, imaging) that are independent of the robot movements. Another way to improve the performance level is to use the Kalman filtering in the sensorial data, since this methodology is very suitable to cope with statistical and stochastic sensor data. Actually, one of the most important applications of the extended Kalman filtering is a simultaneous localization and map building.

In the future we will try to combine the fuzzy logic with this type of filtering or to add other sensors for a better position estimation and control.

References

[1] C. Barten, "Control of mobile robot", 2007.

[2] R. Siegwart, I.R. Nourbakhsh, "Autonomous mobile robots", A Bradford Book, The MIT Press, 2004.

[3] A. Saffioti, "Fuzzy logic in autonomous robot navigation, a case study", 1997.

[4] A. Saffioti, E. Ruspini, K. Konolige, "Using fuzzy logic for mobile robot control", *Int. Practical Application of Fuzzy Technologies*, pp. 185-206, 1999.

[5] R. Choomuang, N. Afzulpurkar, "Hybrid Kalman filter/fuzzy logic based position control of autonomous mobile robot", *International Journal of Advanced Robotic Systems*, vol.2, no.3, pp.197-208, 2005.

[6] L. Astudillo, O. Castillo, P. Melin, A. Alanis, J. Soria, L.T. Aguilar, "Intelligent control of an autonomous mobile robot using type-2 fuzzy logic", *Engineering Letter*, 2006.

[7] http://www.festo-didactic.com

[8] K.M. Passino, S. Yurkovich, "Fuzzy control", Addison Wesley Longman, USA 1998.

[9] E. Ruspini, P. Bonissone, W. Pedrycz, "Handbook of fuzzy computation", Oxford Univ. Press and IOP Press, 1998.

[10] S.R.G. Tourino, A.J. Alvares, "Fuzzy logic control system to the mobile robot motion through web", 2002.