Navigation Control of an Autonomous Mobile Guide Robot

Amir Noabahar Sadeghi Nam

Abstract—Designing and manufacturing of an intelligent, autonomous guide robot, its navigation in indoor environments with the user communication, are the principal objectives of this work. The presented behavior-based maples navigation control architecture performs the robot localization, obstacle avoidance, wall following, and path planning to steer it from any initial pose to any assigned location, by employing the onboard sensors. The research aims to reveal the pros and cons of the behavior-based maples navigation against map-based navigation control. Also a graphical interface is designed and presented to interact with the user. In order to fix the encountered data transmission problems between the electrical hardware, some proper techniques are presented. On the other hand, in addition to some electrical and mechanical modifications, two control techniques are proposed to improve the wheels skidding, and the crooked drive mechanical deficiencies.

Index Terms — Autonomous Robots, Behavior-Based Control, Control Architecture, Guide Robots, Human-Robot Interaction, Maples-Based Control, Safe Navigation

1 Introduction

Service or assistant robots are mostly autonomous robots, which always emphasize providing an appropriate service and giving assistance, instead of focusing on the operating with accurate performance. These kinds of robots with different features and appearances are designed and produced to assist and serve needy people along with some other innovative functions at various places as commercial areas, hospitals, shopping malls, etc. The guide robots as a kind of assistant robots, can guide visitors who want to go to a certain place and then explain them some information of their based on the predefined data. The autonomous guide robot task can be categorized in two main separate functions: navigation and interaction. The first function is dealing with how to navigate the robot in a dynamic and crowded environments safely, and the second function is considering how to interact the robot with the visitor effectively. On the other hand, the indoor navigation issue can be decoupled as map-based and non-map-based navigation. In map-based navigation, the robot has a predefined map built in the robot to use for locating itself, but in maples navigation, the robot has no any predefined map. Here, the location of the robot are specified by employing the real-time data from the onboard sensors. The researchers have presented many complete integrated systems, regarding the autonomous guide robots in two mentioned issues, navigation in the indoor or outdoor environments [4], [6], [7], [8], [10], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25]], and human-robot interaction [4], [5], [7], [8], [9], [11], [13], [17], [21].

Design methodology and system architecture of autonomous robots have been addressed in some literature. Design and mathematical models of an automated guide vehicle, has presented in [1]. The vehicle kinematics and dynamics modelling are introduced, a navigate control system is developed, and then the robot assembly, and the test results are explained. In an article [2], system architecture of an intelligent guide robot is presented. It has some abilities to voice communication, face recognition and navigation through a touch pad based graphical interface. In order to increase the autonomy and interactivity of the mobile robots, a fully autonomous tour guide robot is designed, manufactured and presented in [3]. Developing of a guide robot to interact with the human interaction is presented in [4]. The robot can detect multiple persons around it and select one as the main user, then it guides the user towards the destination along with some projected useful information on a surface. Human-robot voice communication is considered to develop design methodologies of a tour-guide robots in [5]. A security and patrol robot to navigate and also maintain the public is manufactured and presented in [6]. Then a controller running on real time windows operation system is designed and applied on the robot. Some literature have presented development of the robots for different applications. An interactive tour guide robot is utilized to safe navigate in unmodified environments, and to interact with the user in [7]. In another article, an autonomous guide robot is presented which communicates with the robot through the web-based interfaces [8]. The robot, is able to travel to different places in a building, and broadcast its camera images during. A semi-autonomous robot is developed to explore possible robot tasks in daily life in a shopping mall [9]. The speech recognition difficulties in a real environment and also unexpected situations are studied using this robot. In a manuscript, a robot is presented which is able to recognize the open doors and move through it employing an image processing system [10]. Then, if the door is open, the position of the mobile robot with respect to the open door is determined. A shopping assistant robot, which can serves people in a mall is conceptually designed and presented in [11]. The proposed controller is a fuzzy controller which can implement a collision-free guiding and following tasks. An outdoor tour guide robot which can autonomously guide visitors and provide useful information of the visited places is developed in [12]. The reliability and safety of the robot is implemented by sensor fusion, applying path planning, and obstacle avoidance algorithms. An article studies a framework

• *Amir Nobahar Sadeghi Nam, PhD. Degree in electronics and electrical engineering department in Atilim University, Ankara, Turkey*
E-mail: amir.nobahar@gmail.com

for a mobile robot to have the flexibility of deciding the way that the user wants to be guided [13]. The presented framework monitors and adapts to the user and carries out appropriate purposes. The presented guide-guard robot in [14], can perform the human detection and tracking, motion planning, and remote supervise. In addition, it has robot arms, and a power estimation to avoid the robot shout down suddenly. A hybrid-structure robot is designed and presented in [15], which can pass the non-flat grounds. The robot application is performing the home security tasks. Another security robot is developed in [16], which can detect abnormal and hazardous cases and transmit a notification to client computer. Some other literature, have been presented the navigation systems in autonomous robots. The study [17] addresses an autonomous navigation system along with a human robot interaction system for an indoor tour guide robot. The presented navigation system consists of global localization, path planning, local goal-seeking, obstacle avoidance, and behavior fusion. Trajectory planning of a guide robot employing distance-type fuzzy reasoning method and quantified knowledge is described in [18]. The mentioned guidance knowledge is implemented applying the production rules based on the linguistic variables. A tour guide robot is employed to study a smooth and efficient obstacle avoidance method [19]. As the author mentions, the obstacle avoidance control loop should operate fast in order to its execution in real-time, and leave enough processing resources to localization, sensor acquisition, motor control, etc. Navigation of mobile robots through crowded environments such as shopping places, and airports, is studied in [20]. Based on the sample paths, the presented approach employs inverse reinforcement learning to learn the behavior of the human-like navigation. In order to navigate of a guide robot, detect visitors and interact with them via voice and touching screen, a novel approach is presented in [21], based on a robust multi-sensor navigation system. A tour guide robot is developed and presented in [22], which performs the navigation task through shape recognition and path planning. The presented robot moves autonomously along a fixed and desired path. Employing a single camera, navigation a mobile robot is studied to avoid the moving obstacles in a research [23]. The block-based motion estimation approach is used to recognize the objects that move near the robot. Detecting of moving objects such as pedestrians, using image processing technology is studied in a resembling paper [24], and its applications in robot field is presented. The presented approach, first estimates the direction of the movement of the objects and then sends an alarm about its own movement, and hence the robot can safely pass by moving objects. In an article [25], in order to identify a specific user in real time, a simple personal identification approach is proposed by applying dress color information.

In this paper, first the system design and manufacturing of a guide robot to safe navigate in an indoor environment, and also human interact is presented. A behavior-based maples navigation control architecture is introduced to localization, obstacle avoidance, wall following, path planning and navigating the robot from any initial position to assigned goal point. In addition, a proper user interface is designed and implemented to interact with the user. Verification of the designed navigation system, and analysis of the behavior-based controllers are fulfilled by implementing some simulated and practical experiments. On the other hand, these results reveals all the electrical and mechanical deficiencies in the navigation system. Some proper techniques, are presented to fix the encountered data transmission problems between the ultrasonic sensors, encoders, user interface and the microcontroller board, with no need to new electrical components and spend more. Also to improve the wheels skidding and crooked drive problems, some electrical and mechanical modifications are performed, and also some control techniques are proposed. The advantages and disadvantages of the vision-based maples navigation control system are studied against the map-based navigation systems. All of these presented control architectures, techniques and contributions to create a flawless navigation system, reveal novelties for this study in the theoretical and practical domains.

2 System Design and Implementation

In order to design an autonomous robot, it is required to integrate many sensors and actuators on a physical base to give the robot the capacity to interact with its environment and perform its specified functions. Much as the employed sensors and actuators impose some limitations on the performance of robot, on-board processing capability is another critical factor required for the robot control. This on-board processing capability strongly effects the structural requirements of the robot. In addition, the robot overall shape as life-sized or human-shaped, has a direct influence on the public environment. Theoretically, the robot can perform its functions independent from its shape, but an appropriate shape is convenient to manufacture the structure and mount the other mechanical and electrical hardware on it. In addition, it strongly affects the complexity of the whole control system.

Classification of all functions which the guide-guard robot should perform, is the first requirement for the system design. Figure 1, represents this classification, which all functions have been categorized in two main sub-category, navigation and interaction. In this section, the design steps of a mobile robot as an integrated system to perform the navigation and communication tasks, are presented.

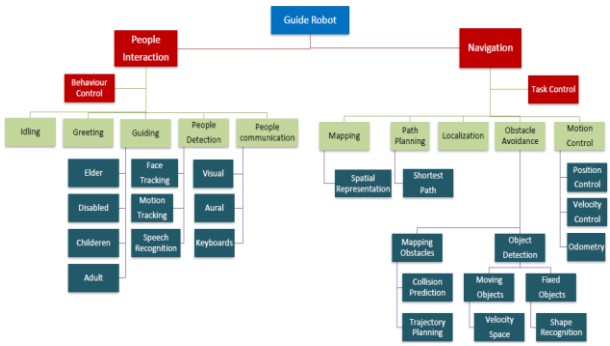


Fig. 1. The General Functions of a Guide Robot

The platform of our designed mobile robot is constructed of a body (chassis framework, floors and walls), two driving and two free-rotating wheels (Omni types), two DC servo motors with integrated encoders, twelve infra-red proximity sensors, four ultrasonic range sensor, a compass, a tablet computer, two microcontroller boards, a motor driver shield and a voltage

regulator. Figure 2 illustrates the arrangement of the sensors, motors and wheels around the chassis of the robot. After mounting and assembling of the mechanical parts and the electronic hardware, the voltage and data lines wiring is implemented. Figure 3, demonstrates the data transmission flow between the subsystems of the electronic hardware.

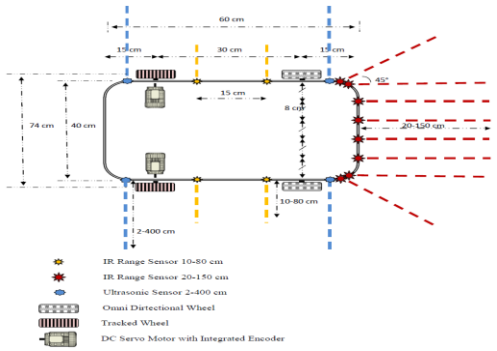


Fig. 2. Arrangement of the Sensors, Motors and Wheels around the Robot Chassis

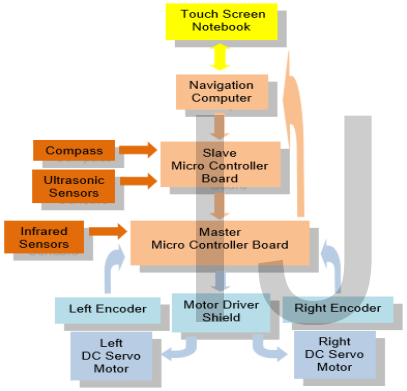


Fig. 3. Data-Transmission Flow between the Electronic Hardware

An *Arduino Mega 2560* microcontroller board is utilized as an embedded system in the robot. Two boards have been employed as the master and slave microcontroller boards (MMCB and SMCB) in our system. A motor driver shield has been mounted on the MMCB to control the DC motors. On the other hand, *MATLAB Simulink* is connected to the microprocessor on the MMCB.

The utilized twelve infrared range (IR) sensors, are placed around the robot, of which six are placed in the front, two on the corners, and four are placed on the sides. Their distance measuring range are 10-80 cm for the sides and 20-150 cm for the front and corner sensors. The raw analogue voltage values from IR sensors is converted to a distance in the MMCB, then it is converted to the points in the robot's and world's reference frame, and then is employed by the planners, as displayed in the figure 4.

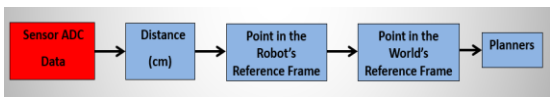


Fig. 4. Converting the IR Sensors Raw Data to be used by the Planners

The employed four ultrasonic sensors which are placed on the sides of the robot, can detect an obstacle in the range of 2-400 cm. We can calculate the distance through the time interval between sending a 10 microsecond input pulse and receiving the echo signal. But this narrow pulse width causes to freeze the model. This problem arises from a big difference between the sample time of the model (10 ms) and this pulse width (10 μs). To solve this problem, both trigger and echo signals should be sent and received by proper codes uploaded on the SMCB. Then the measured distance is sent by TX2 port on SCMB and received by RX2 port on MMCB, as shown in figure 5.

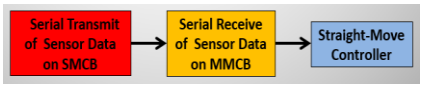


Fig. 5. Transferring the Ultrasonic Sensors Data to be used by the Straight-Move Controller

The robot's heading is determined by the utilized compass as its absolute orientation. The proper data after calculating in the SMCB, is sent by TX3 port on SMCB and received by RX3 port on MMCB, as displayed in figure 6.

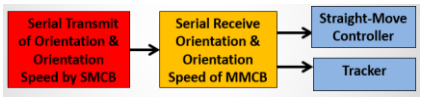


Fig. 6. Transferring the Compass Data to be used by the Straight-Move Controller and the Tracker

The employed servo motor has an integrated quadrature type encoder. It increments or decrements a tick counter depending on the rotating direction of the motor. The number of count per second of the encoders are CPS=12040. This amount is too big to be counted by *DigitalRead* block of the *Arduino* board. In other words, the encoders are too fast, which causes the block to freeze. Others have had the same experience in this problem [26]. One solution can be employing of decoder chips well known as *Quadrature Encoder Interface* (*QEI*) like *LS7266R1*. The results of its usage in a robot project has been presented in [27]. Another suggested solution is employing a high performance signal controller, like *dsPIC 30F4011* [28]. We solved this problem by reading directly from *Atmel* ports (the Microprocessor of the *Arduino Mega 2560*) instead usage of the *Arduino* block. To implement this solution, the utilized encoders are connected to the *External Interrupts* pins on the *Arduino* board. Then a *MATLAB* function block attaches the interrupt service routines (C code) to the two pins to which the encoder is connected. These interrupts service routines update the encoder position, when it rotates. The encoder position is transferred to be used by the odometry. This procedure is illustrated in figure 7.

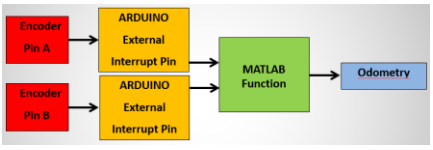


Fig. 7. Transferring the Encoders Data to be used by the Odometry

The final manufactured guide robot is displayed in figure 8. It is worth mentioning that, a tablet computer, is employed as the navigation computer for importing the input's commands and also displaying some information as the outputs. The space work of our guide robot, is as the map platformas shown in figure 9. The coordinates of all the intermeidiary and final goal points are predefined in the robot database.

Fig. 8. The Manufactured Guide

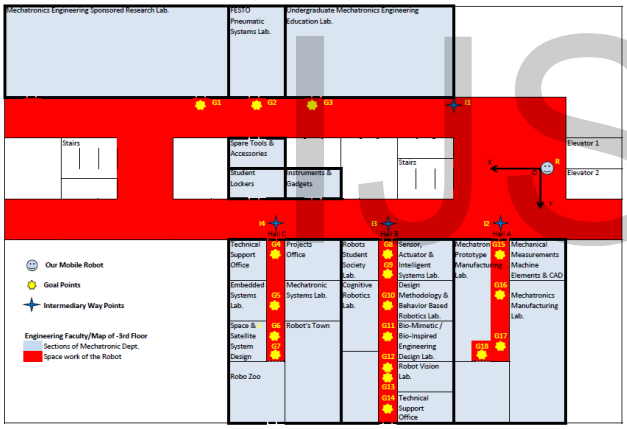


Fig. 9. Intermediary & Final Goal Points on the Map Platform

In order to design control architecture based on the requirements, its main functions to move and sense are determined. These two functions are, transform the controller's outputs to the robot through a supervisor, and keep track of the robot's location. To implement these functions, a supervisor executes the data transferring between the controllers and the robot as illustrated in figure 10.

The user commands are entered by touching the *Push-Bottoms* on the screen of the computer as the graphical user interface, and some information are displayed on it, when the robot reaches to the goal point. Based on the intermediary and goal points on the presented map, a graphical user interface is designed and constructed, as illustrated in figure 11.

A serial communication must be established to transfer the user interface date to the microcontroller. Although it is seemed to be simply done by the USB connection between the computer and the microcontroller, but this idea does not really work. The problem originates from the *Arduino* serial receive block, which does not work with Simulink coder. In other words. The *Arduino* serial block works just during simulation, but after compiling to the execution file it does not perform its task. Others have had the same experience in this regard [29, 30]. One solution as suggested can be usage of the *byte pack* block before the *Arduino* serial block. But we could solve this problem in a simple way, by employing the SMCB. This slave board receives the data by serial USB from the user interface and then transmits it serially to the master board. The serial receive block on the master board can catch the serial data.

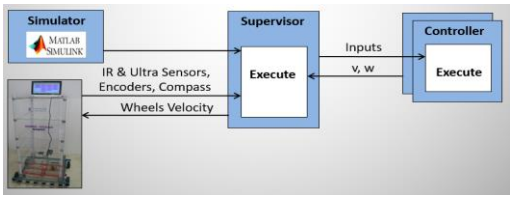


Fig. 10. Schematic of the Control Architecture

Fig. 11. Screenshot of the Graphical User Interface

3 Navigation Control Design

Since the robot has a differential drive system, it should be controlled by determining the rotational speeds of the right and left wheels (ω_r, ω_l), rather than the linear and rotational speeds of the unicycle robot (v, ω). The pose of the robot is composed of its position and orientation (x, y, φ), well konown as the odometry. The dynamics of the unicycle robot and the differential drive are presented in (1), and (2) respectively, where R is the radius of the wheels and L is the distance between the wheels. The current pose of the robot is achieved by the odometry as (3), where D_r and D_l are the distance which right and left wheels have been traveled.

In the next step, employing a classic PID controller, it is tried to make the actual motor speed match the desired value. In order to design the control systems, firstly the dynamic modeling of the prototype setup is extracted. As it is displayed in figure 12, the system model is extracted by system identification based on the dynamics of the differential-drive, and then the speed and position controllers are designed. The design of the behavior-based controllers are fulfilled based on the unicycle model.

$$\begin{aligned} \dot{x} &= v \cdot \cos\varphi \\ \dot{y} &= v \cdot \sin\varphi \\ \dot{\varphi} &= \omega \end{aligned} \tag{1}$$

$$\begin{aligned}\dot{x} &= \frac{R}{2}\cdot(\omega_r+\omega_l)\cdot\cos\varphi\\ \dot{y} &= \frac{R}{2}\cdot\left(\omega_r+\omega_l\right)\cdot\sin\varphi\\ \dot{\varphi} &= \frac{R}{L}\cdot\left(\omega_r-\omega_l\right)\end{aligned} \quad (2)$$

$$\begin{aligned}x &= x_{initial}+\frac{D_r+D_l}{2}\cdot\cos\varphi\\ y &= y_{initial}+\frac{D_r+D_l}{2}\cdot\sin\varphi\\ \varphi &= \varphi_{initial}+\frac{D_r-D_l}{L}\end{aligned} \quad (3)$$

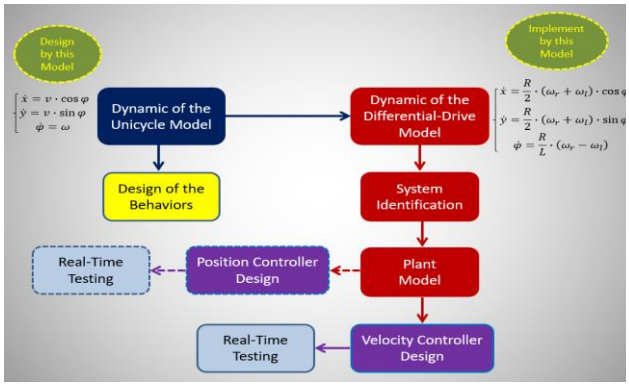


Fig. 12. Design and Implement of Controllers employing the Unicycle and Differential-Drive Dynamics

3.1 System Dynamic Modelling

We can generally model a plant in two methods, mathematical modeling which needs detail knowledge of all plant components dynamics, and data-driven modeling which requires the input-output data measurement known as system identification [31]. Since the first method can practically get challenging and also by applying so many assumptions and approximations in the model, it leads to a non-accurate model. So we decided to extract the system model using the system identification. To this aim and also extract an accurate system model, we prefer to measure the outputs when the robot is travelling. As it is displayed in figure 13, the input signal is the desired speed of the motor (in rpm) and the position of the motor (in degrees) is measured as the output.

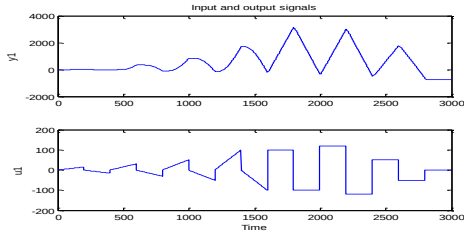


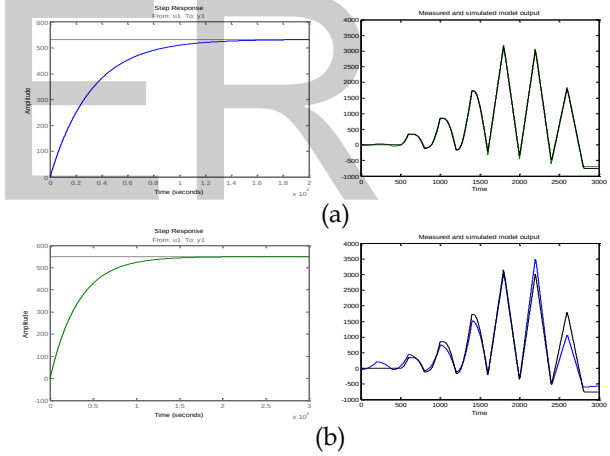
Fig. 13. Schematic of the Control Architecture

Now, using this measured input-output data, it is tried to construct the mathematical model of the plant, by identification of the linear and non-linear models. The model estimation procedure is implemented employing linear models with different number of poles and zeros, and nonlinear models with different number of inputs and outputs. Table 1, displays the estimated linear and nonlinear models. Considering the simplicity of the estimated model and also fit to estimation data, the linear model "STF1" and nonlinear model "SHW1" are chosen as the best models, which can simulate our setup. The estimated linear model can be represented by a transfer functions as (4). On the other hand, the estimated nonlinear *Hammerstein-Wiener* model, can be applied by an "*IDNLHV*" model block, which simulate it for time-domain input and output data in the *Simulink*. Figure 14 displays the time signals of the measured and simulated model output, and also the step response of these two estimators.

$$\mathrm{STF1} = \frac{0.02638\ (\pm 0.003287)}{s^2-1.843(\pm 0.01973)s+0.8431(\pm 0.01973)} \quad (4)$$

TABLE 1
MODEL ESTIMATION EMPLOYING LINEAR AND NONLINEAR MODELS

Model	Name	Number of Poles / Inputs	Number of Zeros / Outputs	Fit to Estimation Data
Linear	**STF1**	2	1	**76.31%**
Linear	STF2	2	2	76.40%
Linear	STF3	3	1	76.38%
Linear	STF4	3	2	76.39%
Linear	STF5	3	3	76.40%
Nonlinear ARX	SARX 1	2	2	32.00%
Nonlinear ARX	SARX 2	4	2	72.00%
Nonlinear ARX	SARX 1	3	3	13.00%
Nonlinear H-W	**SHW1**	10	10	**97.37%**
Nonlinear H-W	SHW2	12	12	96.59%



(a)

(b)

Fig. 14. Time Signals of the Measured and Simulated Model Output and Step Responses of (a) STF1 (b) SHW1 Estimators

3.2 Behavior-Based Navigation Control

The presented behavior-based maples control architecture, performs the navigation task including localization, obstacle avoidance, path planning and moving the robot from any initial position to assigned goal point. These navigation functions are implemented by the control architecture in a maples and unknown environment. However, to realize this, we have to design more complicated controllers, employ more hardware, and consequently pay more, in comparison with the map-based systems.

The goal points are given to the planner state machine to produce a proper output vector. Then this vector is applied as the reference trajectory to the tracker, where the rotational speeds of the right and left motors (v_r, v_l) are produced [32], as illustrated in figure 15.

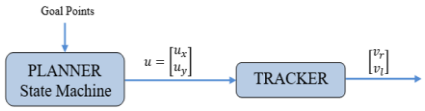


Fig. 15. The Planner and the Tracker

3.2.1 Planner

Since the controller must respond to a dynamic and unknown environmental conditions, so the whole control task should be divided, rather than constructing a complicated controller. In other words, the control task is considered as the behaviors such as Go-to-Goal (GTG), Obstacle-Avoidance (AO), and Wall-Following (FW) behaviors. Then switching among the behaviors as a hybrid navigation system, in response to the changes of the environment is simpler than have a complicated controller. This is known as behavior-based robotics [32, 33].

- **Go-to-Goal Behavior (GTG)**

This behavior steers the robot towards the goal point. The reference trajectory is simply calculated as (5), using the robot current location (x, y) and goal location (x_G, y_G). The GTG vector is displayed in figure 16.

$$u_{GTG} = \frac{\begin{bmatrix} x_G - x \\ y_G - y \end{bmatrix}}{\left\| \begin{bmatrix} x_G - x \\ y_G - y \end{bmatrix} \right\|} \quad (5)$$

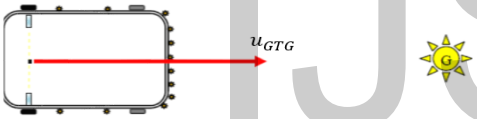


Fig. 16. Go-to-Goal Vector

- **Avoid-Obstacles Behavior (AO)**

This behavior keeps the robot away from the obstacles. Actually, this capability increases the robot quickness, facing with the moving obstacles in a maples and unknown environment. The strategy for this behavior is dedicating a vector to each of twelve IR sensors $(u_1, u_2, \dots, u_{12})$, weighing each of these vectors according to the sensors importance, (for instance, the front sensors are typically more important, for obstacle avoidance while moving forward), summing the weighted vectors to form a single vector and finally computing the AO vector as (6). The AO vector is illustrated in figure 17.

$$u_{AO} = \frac{\alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_{12} u_{12}}{\left| \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_{12} u_{12} \right|} \quad (6)$$

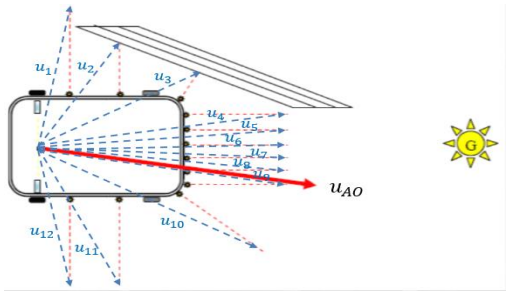


Fig. 17. Avoid-Obstacles Vector

- **Blended Behavior**

This behavior is blending of GTG and AO behaviors instead of hard switching of them. To implement this, each of GTG and AO vectors are weighed according to their importance with ratio α, and then linearly combined into a single vector as (7). It is necessary to tune the ratio α carefully to get the best balance between these two vectors. This blended AO-GTG vector is displayed in figure 18.

$$u_{AO-GTG} = \frac{\alpha \cdot u_{GTG} + (1-\alpha) \cdot u_{AO}}{\left| \alpha \cdot u_{GTG} + (1-\alpha) \cdot u_{AO} \right|} \quad (7)$$

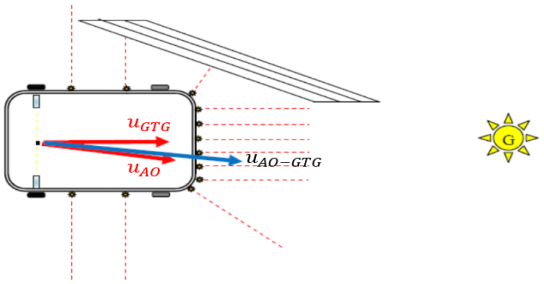


Fig. 18. Blended Vector

- **Follow-Wall Behavior (FW)**

This behavior makes the robot follow the boundary of an obstacle or a wall. To implement this strategy first, the vector, u_{FWt} is constructed which estimates a part of the wall next to the robot employing the IR sensors on the sides of the robot. Considering the below figure 19, to compute the FW vector, the following computations should be implemented as presented in (8) till (12). In these equations, d_{FW} is the distance which the robot should maintain from the wall.

$$u_{FWt} = u_3 - u_2 \quad , \quad \hat{u}_{FWt} = \frac{u_{FWt}}{|u_{FWt}|} \quad (8)$$

$$u_R = \begin{bmatrix} x \\ y \end{bmatrix} \quad , \quad u_P = u_2 \quad , \quad u_{PR} = u_P - u_R \quad , \quad \hat{u}_{PR} = \frac{u_{PR}}{|u_{PR}|} \quad (9)$$

$$u_{FWp} = u_{PR} - (\hat{u}_{PR} \cdot \hat{u}_{FWt}) \cdot \hat{u}_{FWt} \quad , \quad \hat{u}_{FWp} = \frac{u_{FWp}}{|u_{FWp}|} \quad (10)$$

$$u'_{FWp} = u_{FWp} - d_{FW} \cdot \hat{u}_{FWp} \quad (11)$$

$$u_{FW} = d_{FW} \cdot \hat{u}_{FWt} + \left(u_{FWp} - d_{FW} \cdot u'_{FWp} \right) \quad (12)$$

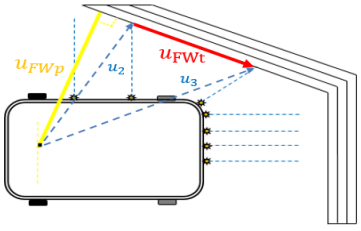


Fig. 19. Two vectors in FW Behavior

3.2.2 Hybrid Navigation System

In order to design logic for supervisory control, fault management, and control task scheduling, state-flow environment is used [34]. In this environment, the decision logic is designed based on the state machines and flow charts. Taking advantages of the capabilities of this environment, we

combine all four behaviors in a single hybrid navigation system. To construct this hybrid system, six below events should be firstly defined, and then we can combine the behaviors as the states in a single hybrid navigation plan, as illustrated in figure 20.

1. At-Goal-Event (AGE): occurs when the remaining distance to the goal point is smaller than a predefined value (d_0).

$$dtg = \sqrt{(x_G - x)^2 + (y_G - y)^2} \quad , \quad dtg \leq d_0 \qquad (13)$$

2. At-Obstacle-Event (AOE): occurs when the measured distance to the obstacle by each IR sensor is smaller than a predefined value (SR_{AOE}). This value should be different for the side, corner, and front sensors.

3. Progress Event (PE): occurs under the condition presented in (14). In this equation, d_P is the closest distance the robot has progressed towards the goal, and ε is a small enough constant.

$$dtg \leq d_P - \varepsilon \qquad (14)$$

4. Sliding Event (SE): derermines whether the robot should continue to FW or switch back to the GTG behavior. Considering (15), this event occurs, when ρ_1 & $\rho_2 > 0$.

$$\begin{bmatrix}\rho_1 \\ \rho_1\end{bmatrix} = [u_{GTG} \quad u_{AO}]^{-1} \cdot u_{FW} \qquad (15)$$

5. Unsafe Event (UE): occurs when the measured distance to the obstacle by each IR sensor is smaller than a predefined value (SR_{UE}). This value should be different for the side, corner, and front sensors.

6. Double Unsafe Event (DUE): occurs when the measured distance to the obstacle by each IR sensor is smaller than a predefined value (SR_{DUE}). This value should be different for the side, corner, and front sensors.

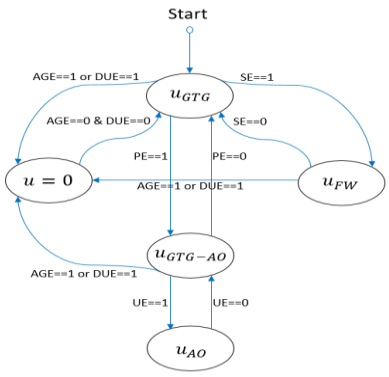


Fig. 20. Behaviors (States) in a single Hybrid System

3.2.3 Trackers

In order to design a tracker for our differential-drive robot, the output from the planner should be convert as the input reference trajectory to the rotational speeds of the right and left motor as presented in (18), concluded from (17). As it is clear, a classic PID controller has been employed to achieve the robot's rotational speed (ω), using the desired robot's angle (φ_d).

$$u = \begin{bmatrix}u_x \\ u_y\end{bmatrix} \qquad (16)$$

$$\varphi_d = \tan^{-1}\left(\frac{u_x}{u_y}\right), \omega = PID(\varphi_d - \varphi), v = \sqrt{{u_x}^2 + {u_y}^2} \qquad (17)$$

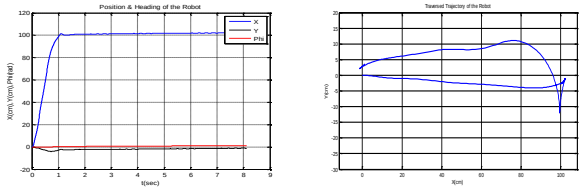
$$\omega_r = \frac{2v + \omega L}{2R} \quad , \quad \omega_l = \frac{2v - \omega L}{2R} \qquad (18)$$

It is worth mentioning that, there are two critical limitations for the utilized motors. They have a maximum rotational speed $\omega_{r\&l(max)} = 9\, rad/sec$, and they stall at low speeds $\omega_{r\&l(min)} = 0.7\, rad/sec$. So if (v, ω) are both small, we have to scale up v, and if they are both large, we have to scale down v to make ω possible.

4 Experiments and Results

In order to test and validate the robot navigation and interaction system along with the designed controllers, three kinds of experiments are implemented. The first experiment is performed on the simulated system, to analyze the efficiency of the controllers, especially the GTG controller. The second test is implemented on the robot but not during its movement, to analyze the efficiency of the GTG, AO and Blended controllers. The third experiments are conducted for dozens of times in the field on the robot with different goal points, to complete and improve the efficiency of the GTG, AO, Blended and FW controllers. The mechanical behavior of the components and whole of the system performance are checked out during the third experiments.

In the first experiment, employing the plant model achieved from the system identification, the GTG behavior is checked in the Simulink environment. The heading and position of the robot, when it tries to reach from the initial point at (0, 0) to the goal point at (100, 0), are extracted as the time plot illustrated in figure 21. The traversed trajectory of the robot, when it moves towards the goal point and then comes back to the initial point, is displayed in the second graph.

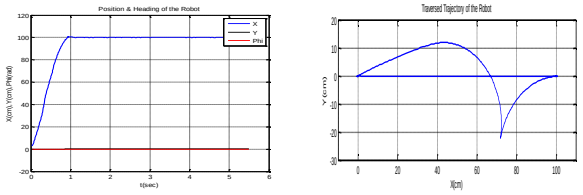


(a) (b)

Fig. 21. (a) Position, Heading, (b) Traversed Trajectory of the Robot to reach the Goal Point at (100, 0) and come back to the Initial Point in the Simulink Environment

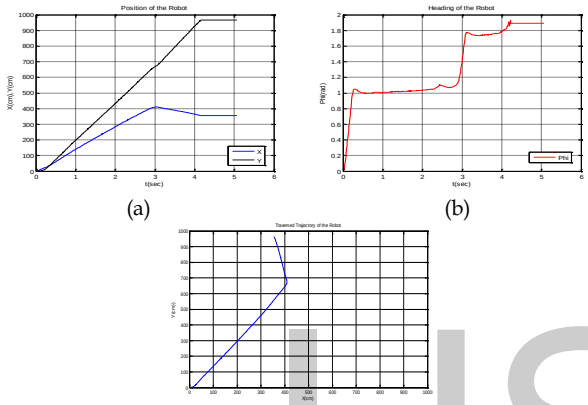
In the second experiments, first the same test is repeated on the actual robot, when the robot powered wheels are not on the ground. By this way, performance of the real navigation system is tested. First a GTG behavior is implemented. The heading and position of the robot, when it tries to reach to the goal point, are derived as the time plot displayed in figure 22. Then we opt another goal point, with an intermediary point on the way. The robot tries to reach the goal point at (354, 956) through the way point (404, 655). The heading and position of the robot, are extracted as the time plot illustrated in figure 23 in two separated graphs, which is followed by the traversed trajectory in third graph. Whole the navigation system shows an efficient performance successfully. In another test, our goal point is at

(2000, 0), but there is an obstacle on the way. The performance of the GTG, and AO controllers are checked in this test. Figure 24 shows the heading and position of the robot and the traversed trajectory.



(a) (b)

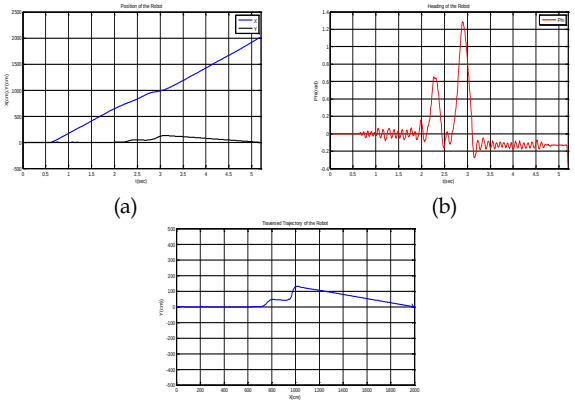
Fig. 22. (a) Position, Heading, (b) Traversed Trajectory of the Robot to reach the Goal Point at (100, 0) and come back to the Initial Point in the Actual Robot



(a) (b)

(c)

Fig. 23. (a) Position, (b) Heading, (c) Traversed Trajectory of the Robot to reach the Goal Point at (469, 1261) through the Intermediary Way Point at (469,552) in the Actual Robot



(a) (b)

(c)

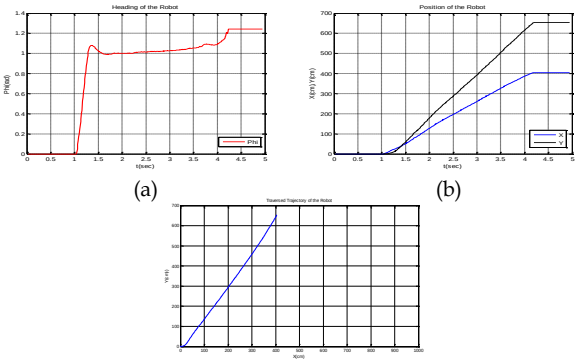
Fig. 24. (a) Position, (b) Heading, (c) Traversed Trajectory of the Robot to reach the Goal Point at (2000, 0) with an Obstacle on the Way in the Actual Robot

Eventually we start to test the robot in the field as the third experiments. After a couple of tests, we figured out that the electronic hardware operates properly based on the control behaviors. But two undesirable physical problems occur. The first problem is that the powered wheels skid at the time when the robot wants to move. Actually, at initial time the rotational speed of the motors increases and causes the wheels to slide on the ground. This undesirable problem concludes to reach a wrong final position. To solve this problem, the rotational speed of the motors should change smoothly, in other words, it should have ramp shape rather than step shape. In addition, the type of the powered wheels are improved with the wheels that are made from anti-slide material.

The second problem is that the robot dose not move straight, which leads to have a big final error at goal point. The crooked-drive arises from different physical problems. Generally, applying the same power to each wheel in any vehicle does not result straight moving. To solve this problem, we design a proper controller. In this controller, a classic PID controller uses the difference of motors speed to create the rotational speed difference that must be applied to the robot controller. With the help of this controller, we can be sure that the two wheels are rotating at the same speed. But there could be a couple of other physical problems. The wheels might still be somewhat different diameters, the gravity center may not exactly be in the middle, the robot might not be symmetrical, pairs of rear and front wheels or one of them may not be parallel with each other or with the robot's side, etc. Because of all these unknown problems, there may still be small deviation. So we tried first to improve the above problems mechanically, as much as feasible. Then, to improve the situation more, we employ the ultrasonic sensors to eliminate the deviation and keep the robot move straight as the first solution. The ultrasonic sensors on the robot sides measure the distance to the parallel wall to calculate the robot heading and then a proper controller improves the heading. But if there is no wall, the reliable solution is having the robot heading by employing the compass. Applying this solution has another benefit. However we have applied a controller to overcome wheels skid, but if there still exists this problem, the heading of the robot will not be derived wrongly. After designing and employing all the above mentioned controllers to the control architecture of the navigation system, the robot tests in the field are continued. The experiments are implemented with different goal points and with or without intermediary way points. Table 2 displays the implemented tests with different conditions in the field, and figures 25 till 32, show the results as the time plots of robot heading and position and the traversed trajectory. As it can be resulted from the plots, the final robot position differs slightly from the desired position. There are some solutions to tune the robot final position. This can be simply done by guiding the robot to the desired position. The final guidance can be implemented employing different methods as, placing beacons, markers, bar codes, painting lines, etc. near the goal points. One reliable and accurate method, is usage of the indoor positioning systems [35]. In these systems, the exact robot position is extracted using by radio waves, magnetic fields, acoustic signals, or any other sensory information.

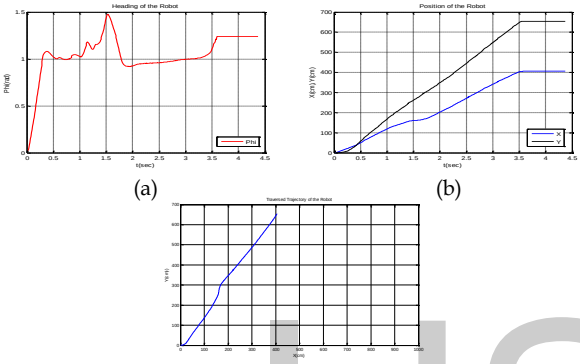
TABLE 2
IMPLEMENTED TESTS WITH DIFFERENT CONDITIONS IN THE FIELD

Test No.	Goal Point	Intermediary Way Point	Fixed Obstacle	Movable Obstacle	Wall
1	(404,655)	No	No	No	No
2	(404,655)	No	Yes	No	No
3	(404,655)	No	No	Yes	No
4	(354,956)	(404,655)	No	No	Yes
5	(1360,830)	No	Yes	Yes	Yes
6	(1280,1298)	(1360,830)	Yes	No	Yes
7	(845,-450)	No	No	Yes	No
8	(2230,-300)	(845,-450)	Yes	Yes	Yes



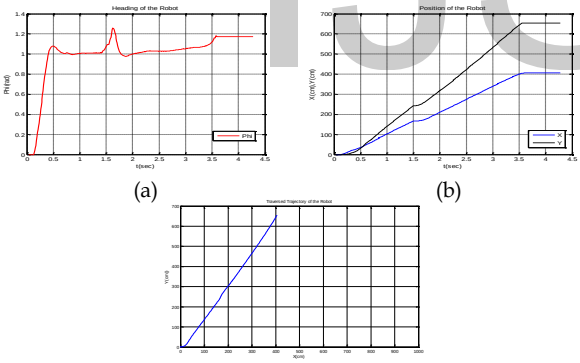
(a) (b) (c)

Fig. 25. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 1



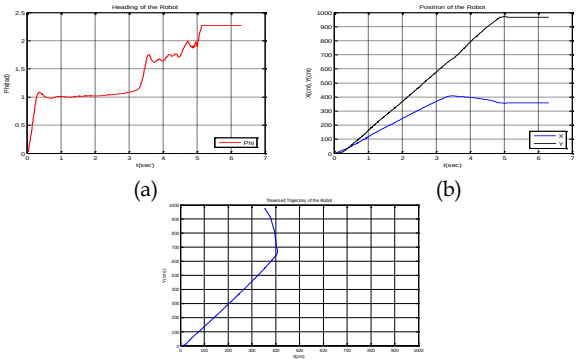
(a) (b) (c)

Fig. 26. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 2



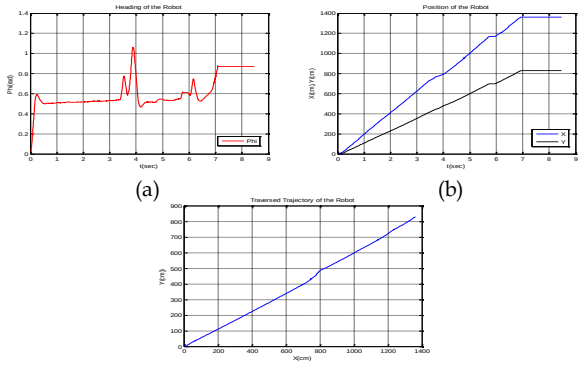
(a) (b) (c)

Fig. 27. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 3



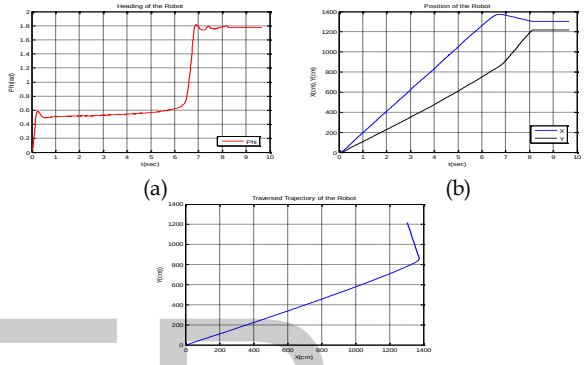
(a) (b) (c)

Fig. 28. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 4



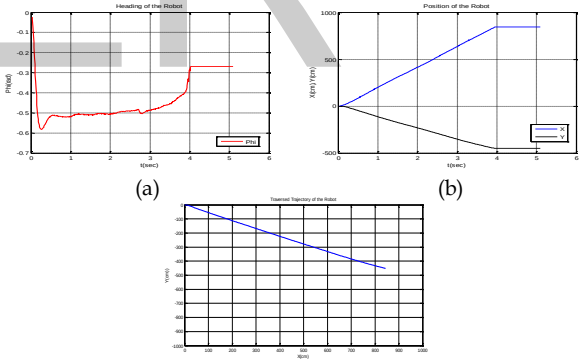
(a) (b) (c)

Fig. 29. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 5



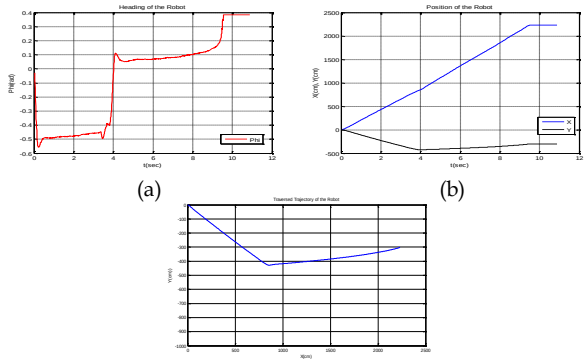
(a) (b) (c)

Fig. 30. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 6



(a) (b) (c)

Fig. 31. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 7



(a) (b) (c)

Fig. 32. (a) Position, (b) Heading, (c) Traversed Trajectory in Test 8

5 Discussion and Conclusion

This study is presented the navigation of a mobile robot in indoor environments along with the user communication. The presented behavior-based maples navigation control architecture, composes of localization, obstacle avoidance, wall following, path planning and steer the robot towards the goal points. In addition, some effective solutions are presented to the revealed electrical and mechanical deficiencies in the navigation system. In the designing and manufacturing of the robot electrical hardware, the encountered data transmission problems between the ultrasonic sensors, encoders, user interface and the microcontroller board, are fixed with no need to new electrical components and spend more. On the other hand, to improve the wheels skidding and crooked drive problems, some electrical and mechanical modifications are implemented, and also some control solutions are proposed.

The research also aims to determine the advantages and disadvantages of the behavior-based maples against map-based navigation. In the presented behavior-based control architecture, the onboard sensors has the main role to perform its functions. Such a vision-based navigation has some pros compared with the map-based navigation. The major advantage is no need for any map to navigate. The robot is able to navigate to any position in a maples unknown environment. Its higher efficiency and quickness, facing with the moving obstacles, is the other advantage. Despite these pros, maples navigation also has some drawbacks. More complicated control architecture, need more hardware, and high costs can be counted as these cons.

In order to validate the designed navigation system, and analysis of the behavior-based controllers some simulated and practical experiments are performed. Based on the results of the experiments, it is revealed that the main trouble to have a flawless navigation, originates from mechanical deficiencies. The stuff of the powered wheels, the gravity center of the robot, the structure symmetry, and the parallel wheels, etc. are some of the major issues which we should pay more attention in designing and manufacturing of the robot mechanical structure. In this case, besides the presented perfect behavior-based control architecture, we can have a flawless navigation system.

References

[1] Archila, John Faber, and Marcelo Becker. "*Mathematical models and design of an AGV (Automated Guided Vehicle).*" 8th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2013.

[2] Oh, SeungSub, et al. "A *system architecture for intelligent building guide robot PHOPE*." 4th International Conference on Autonomous Robots and Agents, ICARA, IEEE, 2009.

[3] Tomatis, N., et al. "*Building a fully autonomous tour guide robot: Where academic research meets industry*." Proc. Int. Symp. on Robotics. 2002.

[4] Sasai, Takuya, et al. "*Development of a guide robot interacting with the user using information projection—Basic system.*" IEEE International Conference on Mechatronics and Automation (ICMA), 2011.

[5] Prodanov, Plamen J., et al. "*Voice enabled interface for interactive tour-guide robots.*" IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 2. IEEE, 2002.

[6] Dehuai, Zeng, Xie Cunxi, and Li Xuemei. "*Design and Implementation of a security and patrol robot system.*" IEEE Mechatronics and Automation, International Conference, Vol. 4, 2005.

[7] Thrun, Sebastian, et al. "*MINERVA: A second-generation museum tour-guide robot.*" IEEE International Conference on Robotics and Automation Proceedings, Vol. 3, 1999.

[8] Simmons, Reid, et al. "*Lessons learned from Xavier*." Robotics & Automation Magazine, IEEE 7.2 (2000): 33-39.

[9] Kanda, Takayuki, et al. "*An affective guide robot in a shopping mall*." Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction, 2009.

[10] Cokal, Erhan, and Abdulkadir Erden. "*Development of an image processing system for a special purpose mobile robot navigation*." IEEE Fourth Annual Conference on Mechatronics and Machine Vision in Practice, 1997.

[11] Shieh, Ming-Yuan, et al. "*Design and Implementation of a Vision-Based Shopping Assistant Robot*." IEEE International Conference on Systems, Man and Cybernetics, SMC'06. Vol. 6, 2006.

[12] Chiang, Kuo-Hung, et al. "*Multisensor-based outdoor tour guide robot NTUI*." SICE Annual Conference, IEEE, 2008.

[13] Pandey, Amit Kumar, and Rachid Alami. "*A step towards a sociable robot guide which monitors and adapts to the person's activities.*" IEEE International Conference on Advanced Robotics, ICAR, 2009.

[14] Luo, R. C., et al. "*NCCU security warrior: an intelligent security robot system*." IEEE 33rd Annual Conference of the Industrial Electronics Society, IECON, 2007.

[15] Kuo, C. H., et al. "*Remote control based hybrid-structure robot design for home security applications.*" International Conference on Intelligent Robots and Systems, IEEE/RSJ, 2006.

[16] Elnagar, Ashraf, and Leena Lulu. "*A visual tool for computer supported learning: The robot motion planning example*." Computers & Education 49.2 (2007): 269-283.

[17] Tsai, Ching-Chih, et al. "*Autonomous navigation of an indoor tour guide robot*." Workshop on Advanced robotics and Its Social Impacts, ARSO IEEE, 2008.

[18] Mizobuchi, Yoshinobu, et al. "*Trajectory planning method of guide robots for a achieving the guidance*." IEEE International Conference on Robotics and Biomimetics (ROBIO). 2005

[19] Philippsen, Roland, and Roland Siegwart. "*Smooth and efficient obstacle avoidance for a tour guide robot*." ICRA. 2003.

[20] Henry, Peter, et al. "*Learning to navigate through crowded environments*." IEEE International Conference on Robotics and Automation (ICRA), 2010.

[21] Shen, Jiali, and Huosheng Hu. "*Visual navigation of a museum guide robot*." The Sixth World Congress on Intelligent Control and Automation, WCICA, Vol. 2. IEEE, 2006.

[22] Wang, Chun-Chieh, Kuo-Lan Su, and Chih-Teng Shen. "*Implementation of Tour Guide Robot via Shape Recognition and Path Planning*." Fourth International Conference on Innovative Computing, Information and Control (ICICIC), IEEE, 2009.

[23] Kim, Jeongdae, and Yongtae Do. "*Moving obstacle avoidance of a mobile robot using a single camera*." Procedia Engineering 41 (2012): 911-916.

[24] Moriwaki, Katsumi. "*Recognition of moving objects by image processing and its applications to a guide robot.*" IEEE/SICE International Symposium on System Integration (SII), IEEE, 2011.

[25] Sugiyama, Seiji, Kouhei Baba, and Tsuneo Yoshikawa. "*Guide robot with personal identification method using dress color information via KINECT*." IEEE International Conference on Robotics and Biomimetics (ROBIO), 2012.

[26] Arduino forum, Hardware, Interfacing, "*Missing counting encoder*" https://forum.arduino.cc/index.php?topic=22245.0, Visited Date; 30.05.2020

[27] Wilhelm, Eric Jamesson. "*Design of a wireless control system for a laboratory planetary rover*". Diss. Massachusetts Institute of Technology, 1999

[28] Kara, Sertac Emre. "Control *of two wheel self-stabilizing mobile robot with a simple arm*". Mechatronics Engineering, Atilim University, October 2014.

[29] Mathworks, MATLAB Answers, "*Serial Receive block doesn't work with Simulink Coder*", https://www.mathworks.com/matlabcentral/answers/116783, Visited Date; 30.05.2020

[30] Stack Overflow, "*Serial block doesn't work with Simulink coder*", https://stackoverflow.com/questions/21883900/, Visited Date; 30.05.2020

[31] Solomatine, D., Linda M. See, and R. J. Abrahart. "*Data-driven Modelling: concepts, approaches and experiences.* "Practical Hydro-informatics. Springer, Berlin, Heidelberg, 2009. 17-30.

[32] Mathworks, File Exchange, "*Sim.I.am, a robotics simulator*", https://www.mathworks.com/matlabcentral/fileexchange/40860, Visited Date; 30.05.2020

[33] Arkin, Ronald C. "*Behavior-based robotics.*" MIT press, 1998.

[34] Chen, Jian, Thomas R. Dean, and Manar H. Alalfi. "*Clone detection in matlab stateflow models.*" Software Quality Journal 24.4 (2016): 917-946.

[35] Wikipedia, "*Indoor positioning system*", http://en.wikipedia.org/wiki/Indoor_positioning_system, Visited Date; 30.05.2020