Chapter 9

Design and Implementation of a Demonstrative Palletizer Robot with Navigation for Educational Purposes

Dora-Luz Almanza-Ojeda,
Perla-Lizeth Garza-Barron,
Carlos Rubin Montoro-Sanjose and
Mario-Alberto Ibarra-Manzano

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.72872

Abstract

Nowadays, many kinds of robots are used in industries to help in manufacturing or placing objects. However, teaching young people and children about robot design and work can be difficult, turning this into a complicated area for them. This chapter provides a detailed description of the design and implementation of a robotic arm mounted on a mobile robot using the LEGO Mindstorms NXT kit® and the starter kit DaNI 2.0, designed by National Instruments®. The mobile palletizer robot takes a box from place A and navigates in the indoor environment until it reaches a predefined place B. The characterization of the robotic arm is based on a parallel structure considering that the end-effector has only two points to hold the object; the gripper is also built using LEGO®. The robot performs the path computed using an A-star algorithm; moreover, actions like moving up and down, opening and closing the gripper and picking up the box and putting it down are executed by the robotic arm using the central unit of the NXT kit. Each stage of the robot design and implementation is explained in detail using diagrams and 3D graphical views with the aim of illustrating the implementation step by step for educational purposes (mainly for young people or children).

Keywords: mobile robot, robotic arm, parallel structure, path planning

1. Introduction

Robots have been used in applications such as industry, medicine, agriculture, space, education, underwater exploration and many others. Manufacturing processes in industries have

increased considerably the use of robotic arms to automate repetitive and tedious tasks performed under difficult conditions for workers. Moreover, the use of mobile robots in industries also improves the efficiency and accelerates the production process. Mobile robots are equipped with sensors to analyze and interpret information about the environment during navigation [1]. Some applications of mobile robots in industry are as follows:

1. inspection,
2. production control,
3. transport of different kinds of objects by means of palletizing tasks [2].

The palletizing of objects (essentially boxes) in the industry is the process to accommodate boxes on a pallet that is usually performed by fixed robotic arms [3, 4]. In cases when the destination is not fixed, mobile robots are also used to place boxes to a destination. For instance, magnetic strip-guided robots transport the merchandise successfully, albeit only following a linear path. Therefore, one of the best solutions for palletizing objects from an origin to a destination involves the use of robotic arms mounted on mobile robots.

The palletizing task requires a path planning strategy which consists in finding an obstacle-free path for mobile robot navigation from one place to another. Many path planning strategies can be found in the literature for various applications, ranging from video game programming to outdoor autonomous navigation of robots. Path planning methods are based on simplifying the searching area to a 2D matrix in which each element represents a reduced square area of the navigation area (that will be interpreted as a cell) [5]. Thus, each cell can be navigable or not depending on the obstacles on it, and a resulting path is obtained if a set of adjacent navigable cells from the origin to destination is found.

The aim of implementing a Box Palletizing Robot is to encourage young people to explore robotic issues as modular tasks that require design, mathematical modeling, programming and some interest and creativity. In this context, many robotic kits and prototypes have been introduced by different companies such as Vex [6], Arduino, Lego, Zowi from BQ [7], to name but a few. However, in this chapter, we present a palletizer robot that combines two robotic kits: the mobile robot platform Dani from National Instruments (NI) and the Lego Mindstorms NXT 2.0 8547 model used to build the robotic arm. Both robotic kits require basic, medium and high levels of knowledge in line with the final purposes. In our case, we will describe the design, programming and synchronization of both kits.

Additionally, the path planning strategy used in this project is based on the A-star algorithm and basic strategies to control the robotic arm. The characterization of the robotic arm is based on a parallel structure, and it has been built using the LEGO NXT kit. To improve the compatibility between the robotic arm and the robot mobile, the LEGO NXT is programmed on LabVIEW [8], a trademark software of NI, to use the starter kit, which is a robot also distributed by NI. An Ethernet connection is used for communication between the PC and the mobile robot, while a Bluetooth connection is used for communication with the robotic arm [9].

This chapter describes, in Section 2, the global strategy used to design, implement and program the palletizer robot. The robot implementation and the A-start algorithm are explained in Section 3. Experimental results are presented in Section 4. Finally, Section 5 includes the conclusion and outlines future work.

2. Global strategy for palletizer robot navigation

The palletizer robot proposed here moves a box from place A to place B while navigating and avoiding collisions. To attain this goal, the main tasks involved and tackled here are as follows:

1. Perception of the environment by using sonar and contact sensors
2. Path planning strategy based on an A-star algorithm
3. Performing the robot trajectory and robot interacting with the dynamic obstacles

In general, the strategy programmed and performed by the palletizer robot is described in the block diagram of **Figure 1**. Both programming strategies DaNI and NXT are combined, but the control of the overall task is programmed on the mobile robot DaNI. The action "*compute trajectory to the box*" in the diagram uses the A-star algorithm and receives a predefined map of the environment with all static obstacles on it. This action is programmed on the DaNI robot, and it is performed in two stages; first, the robot moves to the box position and, second, the robot moves to the final pallet. Once the robot performs the first stage and arrives to the box position, the action "*set gripper ready*" involves the configuration and positioning of the gripper to take the box. This action was programmed on the NXT Lego. Once sensors indicate that the gripper has taken the box, the second stage of the trajectory is performed and the robot moves to the "*go to final pallet*" action. The robot locates the gripper and leaves the box carefully on the pallet. Then, the robot goes back to the initial position and the same process starts again if more boxes must be moved. Finally, dynamic obstacles are detected using the sonar sensor during robot mobile navigation and the robot stops if an obstacle is found in its path, and it continues its trajectory when the obstacle is not detected anymore.

2.1. Robot model description

The mobile robot used in this project is the robotic platform called NI LabVIEW robotics Starter Kit®, described in [8], also known as DaNI 2.0, developed by NI. This mobile robot was designed to develop and run algorithms in real time for autonomous system applications and can be programmed on two different languages: LabView or C.

Each wheel of the robot is connected to a DC motor which provides the traction force and stabilization wheel for balancing the robot; the kinematic model and the representation of robot position used in this work is the same as the one presented in [5]. The sbRIO-9632 card

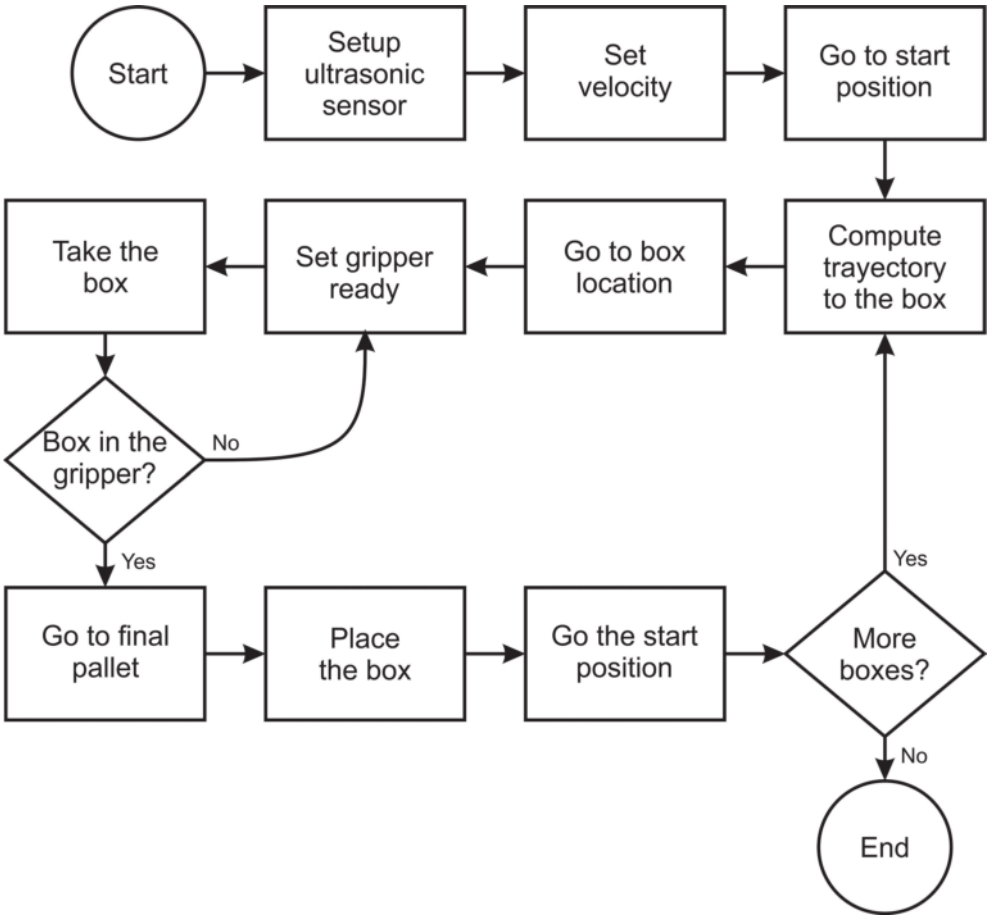

Figure 1. Global strategy for palletizer robot moves a box from place A to place B.

was developed by NI and contains a real-time processor which serves as a main control unit for the robot [10]. In addition, this platform includes a field-programmable gate array (FPGA) Xilinx Spartan-3 which is a reconfigurable device that executes programmed tasks in real time, that is, the active response of the system to external events. For this FPGA, a higher level of programming is possible using the NI LabVIEW® robotics software, which is a graphical language. Programming languages like C, C++ or Java could also be used.

This mobile robot is programmed using an efficient algorithm to cover a trajectory that takes it to the box that needs palletizing. The aim is to illustrate the function of a box palletizer robot in industry. Yet, at this stage, it is only a prototype to show basic functions not involving heavy weights as those handled by an industrial robot.

2.2. Robotic arm with LEGO MINDSTORM NXT 2.0

The Lego Mindstorm is a programmable robotic kit developed by Lego® and introduced for the first time in September 1998. The kit Lego Mindstorm NXT 2.0 provides basic pieces to construct mini-prototypes of robots by means of the assembly of mechanic plastic parts such as wheels, gears and bricks, among others, and electromechanic parts such as motors and different kind of sensors; finally, the robot prototype is programmed in an interactive way. The robots constructed using the Lego Mindstorm kit can simulate the same functionalities as real robots of this kind.

The robotic arm assembled for this project has a degree of freedom; its design is based on a parallel mechanism, and its motion is restricted only to vertical movements (up and down). The NXT brick is the central unit processing for programming robot task using LabVIEW Robotics which is the same graphical language used for programming the mobile robot DaNI. The communication between the NXT module and the PC is via Bluetooth.

2.3. Analysis of the gearbox

The mechanical design of the robotic arm is based on a parallelogram arm (four bars) [11] and an arrangement of gear wheels (called gear train) [12] to transmit turning force and to provide a degree of freedom. The four-bar mechanism consists of two vertical bars of 8 cm in height and two horizontal bars of 6 cm in length. To implement our parallelogram arm, the four-bar mechanism is implemented twice, one for each servomotor used to move the arm up and down. Each motor is finally fixed to the gear train.

A 3D model in Solidworks® [13] of the different ratios of used gears is shown in **Figure 2**. Note that different ratios are used for drive transmission and for the moving arm in accordance with the desired speed ratios, which will be explained below.

The number of teeth on the gear is the most important parameter during gearbox design, because the speed of a final gear train only depends on this parameter, with 24 and 36 teeth being the most common sizes used. With n and Z being the desired angular velocity and the number of teeth of a gear, respectively, both parameters are directly related as:

$$n_1 Z_1 = n_2 Z_2 \tag{1}$$

where index i represents the motor ($i = 1$) and the driven ($i = 2$) gear. This equation provides the number of teeth required to provide a given angular velocity. If the rate Z_1/Z_2 is less than 1, the speed will be reduced. In our case, the gearbox uses eight gears, and we consider $Z_1 = 8$ teeth for gears 1, 3 and 5, and $Z_2 = 24$ for gears 2, 4 and 6, yielding a ratio of *1/3*. The last two gears 7 and 8 are considered as $Z_1 = 16$ and $Z_2 = 36$, respectively, with a ratio of 4/9. **Figure 3** shows the 3D design of the gear train implemented.

Another important parameter during gear train design is the relation between power supply and torque of the servomotor. The Lego servomotor datasheet establishes that

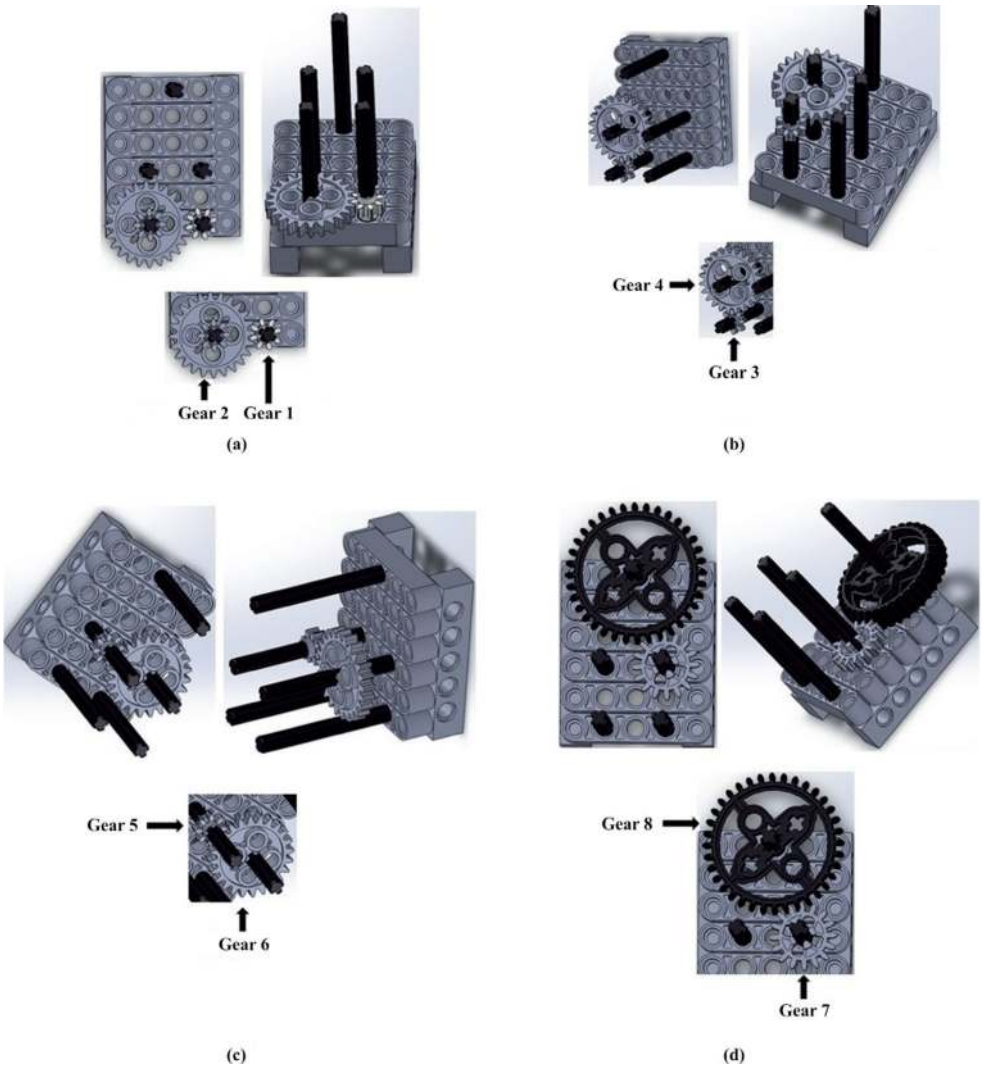

Figure 2. Ratios of the gearbox: three pairs of gears with a ratio of 1/3 are illustrated in (a), (b) and (c); in (d) the final pair of gears 7 and 8 has a ratio of 4/9.

for a power supply of 9 V (i.e., using 100%) the corresponding torque is 19 Ncm, and for 7.2 V (using only 75%) the torque is 16 Ncm [14]. For security reasons, we consider 15 Ncm as the maximal torque value provided by the robotic arm. In addition, **Table 1** shows experimental values of the angular velocity obtained at different values of power supply. Thus, considering a power supply of 75%, the angular velocity of the gearbox is around 95 rpm.

http://dx.doi.org/10.5772/intechopen.72872

On the other hand, with T_1 as the torque of the gearbox, then the power of the robotic arm is obtained by:

$$P = T_1 n_1 \tag{2}$$

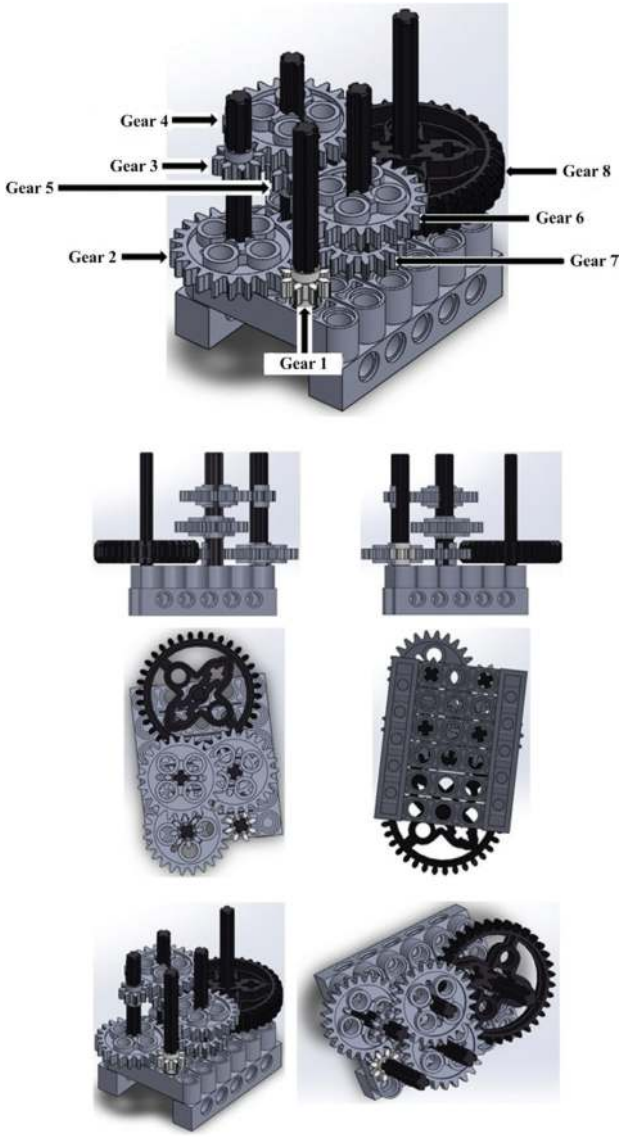

Figure 3. 3D design of the gearbox.

% Power supply	Angular velocity (rpm)
100	135.490
75	95.335
50	61.183
25	25.207

Table 1. Relation between power supply and torque of the servomotor (Lego datasheet [14]).

considering $T_1 = 0.15$ Nm and angular velocity of the motor as $n_1 = 9.9835$ rad/s; thus, the driven power of the gearbox is *1.4975* W.

To obtain the internal velocities along the gear train, we use the torque equation defined as:

$$T_2 = T_1 \cdot \frac{n_1}{n_2} \tag{3}$$

As the angular velocity n_1 is the same as the motor velocity, n_2 is given by:

$$n_2 = n_1 \cdot \frac{Z_1}{Z_2} = 95.335 \cdot (8/24) = 31.7783 \text{ rmp} \tag{4}$$

Then, the torque of gear 2 is obtained using Eq. (3), yielding

$$T_2 = T_1 \cdot \frac{n_1}{n_2} = 0.15 \cdot (95.335/31.7783) = 0.45 \text{ Nm} \tag{5}$$

Following this procedure, **Table 2** shows the torque values for 3–8 gears.

Therefore, torque and angular velocity at the output gear train are 9.1124 Nm and 1.5693 rpm, respectively. If the final torque is divided by the gravity force ($g = 9.81$ m/s^2), then we obtain the mass in kg that the gripper can carry if the robotic arm length were 1 m. In our case, the robotic arm length is 0.20 m, a value that represents one-fifth of the reference value 1 m. By considering that, torque increases in the same factor as the orthogonal length to the applied force decreases, and the final mass that our robotic arm of 0.20 m in length can carry is as follows:

$$0.20 \text{ m} \rightarrow 5(0.9289 \text{ kg}) = 4.6445 \text{ kg} \tag{6}$$

#Gear	Torque (Nm)	Angular velocity (rpm)
3	0.45	31.7783
4	1.35	10.5928
5	1.35	10.5928
6	4.050	3.5309
7	4.050	3.5309
8	9.1124	1.5693

Table 2. Torque and angular velocity of gearbox.

http://dx.doi.org/10.5772/intechopen.72872

3. Implementation of the palletizer robot

The robotic arm assembly requires bricks, girders, angle brackets, gearwheels, three servomotors and four touch sensors included in the Lego kit. Two of the servomotors move the mechanical part of the arm up and down, providing a degree of freedom. The third servomotor is used for closing and opening the griper. One of the touch sensors is at the base of the arm with the aim of sensing the lower position of the arm; similarly, a second touch sensor is located for sensing the higher position that can be reached by the arm. Third and fourth sensors are located on the gripper for measuring the opening and closing degrees controlled by the servomotor. A 3D model of the final robotic arm designed on Google SketchUP® [15] is illustrated on **Figure 4**, and the real robotic arm is shown in **Figure 5**; the gripper consists of four jaws to guarantee that object will be securely held.

The gearbox was designed in line with the required force and velocity to take and move an object from one place to another, without forcing the servomotors. The gearbox uses 8 gears: 3 of 16 teeth, 4 of 24 teeth and 1 of 36 teeth; it was designed following the analysis described in Section 2.3).

Once the robotic arm was built, we slightly modified the DaNI robot structure with the aim of mounting the robotic arm on it. In general, we replaced the rear omnidirectional wheel of

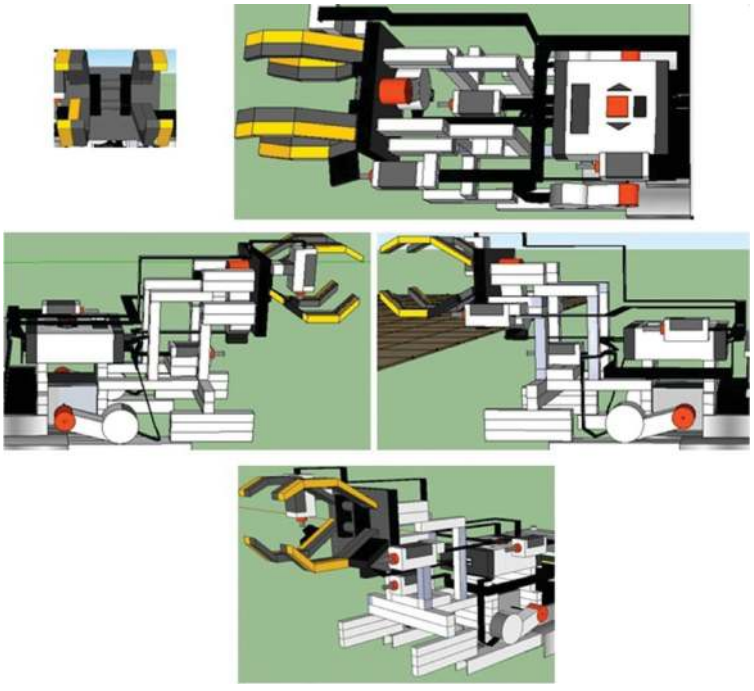

Figure 4. 3D model of the robotic arm designed on Google SketchUP®.

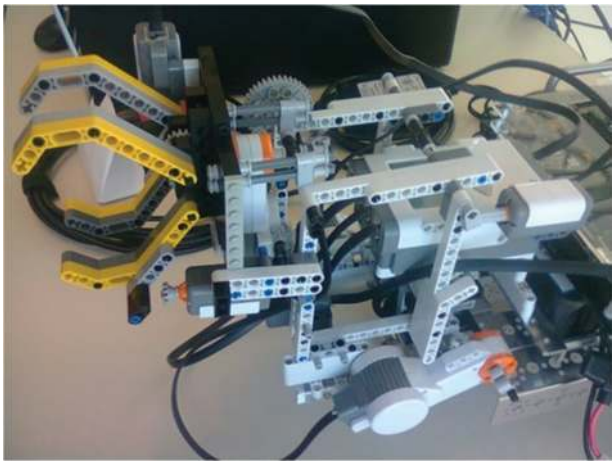

Figure 5. Robotic arm designed using Lego kit.

DaNI for a caster wheel of 1 inch. The wheel was fixed onto the chassis of DaNI creating free space to mount and fix the robotic arm. A 3D model of the mobile robot designed on Google SketchUP is shown in **Figure 6** and the real palletizer robot is shown in **Figure 7**.

3.1. A-star algorithm

The robot trajectory starts in an initial position from which the robot moves to the object location, then it goes to the pallet and, finally, it moves back to the initial position. The final path establishes horizontal and diagonal trajectories that represent the minimal costs to achieve the goal.

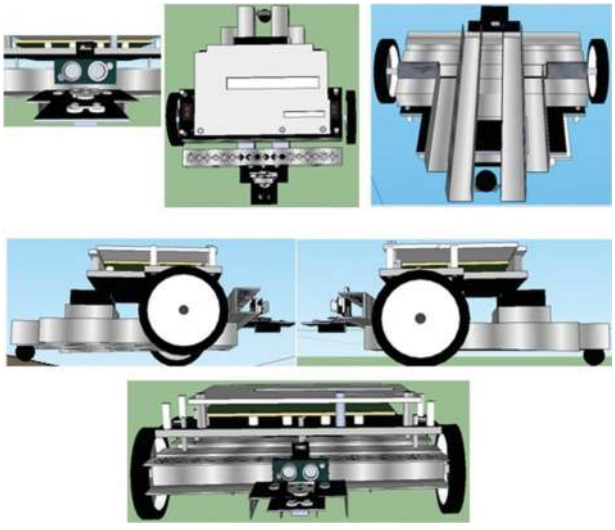

Figure 6. 3D model of the DaNI robot designed on Google SketchUP®.

http://dx.doi.org/10.5772/intechopen.72872

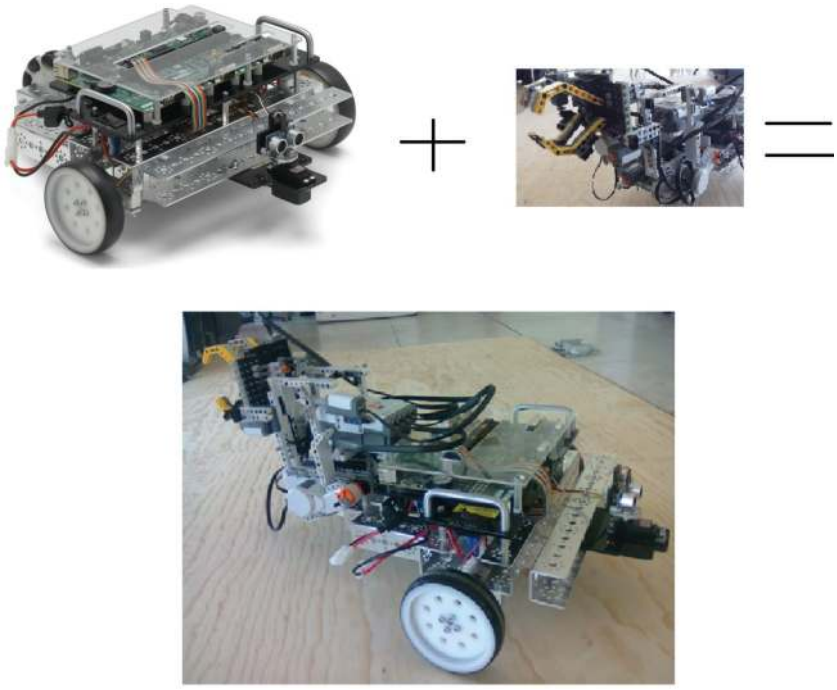

Figure 7. Final palletizer robot.

The planning technique used was the so-called A-star algorithm [16], which basically consists in the research of the best first trajectory that provides the shortest path from all possible roads. The final path is the union of partial movements that the robot must perform to get to the final position, that is, intermediate points before reaching the goal. Here, we use the term of nodes to refer these intermediate points that can be seen also as neighboring or adjacent points to the actual robot position. For each intermediate point, the A-star algorithm evaluates the next trajectories that could be reached based on minimal cost [17]. Thus, a final path guides the robot to a goal position warranting safe navigation.

Figure 8 illustrates a scene that we will use as an example of how to get from position A to B using the A-star algorithm. It is important to point out that two lists are needed to save the adjacent nodes: (1) open list for adjacent nodes that will be compared and (2) close list for nodes that cannot be considered anymore.

The first step to the A-star algorithm is to include the initial node position A to the close list. Thus, the iterative search starts including to the open list all reachable nodes from the initial node A, while unreachable nodes are, for instance, the occupied nodes. In accordance with **Figure 8**, eight nodes were included, depicted as squares and the arrows inside point to their parent node.

Eq. (7) computes the shortest path possible to the goal node B in the fewest moves.

$$F(x) = G(x) + H(x) \tag{7}$$

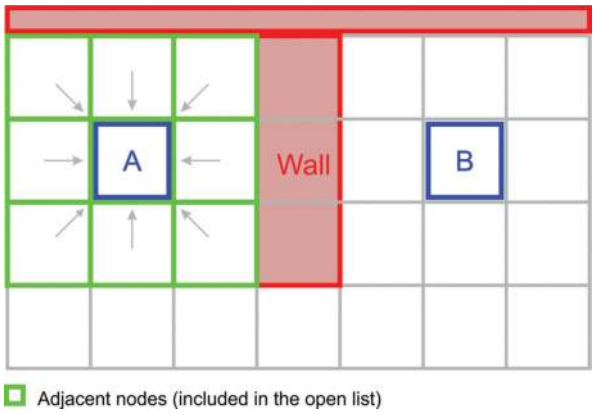

Figure 8. Environment to go from A to B.

where $F(x)$ = cost of the shortest path to the goal, $G(x)$ = cost of the movement from A node to an intermediate node and $H(x)$ = cost of the possible movement to go from an intermediate node to the goal.

Every node on the open list is evaluated using this equation, and the node with the lowest $F(x)$ value is chosen. The value $G(x)$ represents the costs involved in reaching the neighbor nodes. Horizontal or vertical nodes are weighted as 10 (because there is one node to get there). Then, diagonal moves are weighted as 14 because this is the closest without choosing a diagonal. $H(x)$ value is estimated using the Manhattan distance, but many other distances can be used as well. Such distance considers only vertical or horizontal moves to get to the final goal from any intermediate node, ignoring diagonal moves and obstacles on the way. This does not imply that all the environment must be free of moving or static obstacles; it is just the real physical distance between two nodes, after a second weighted value excludes possible paths with occupied intermediate nodes. The punctuation after computing Eq. (7) in our example is shown in **Figure 9(a)**. As the method does not consider obstacles, the shortest distance from B to right node of A is $H = 30$ because this node is reachable in three nodes. Similarly, the upper right

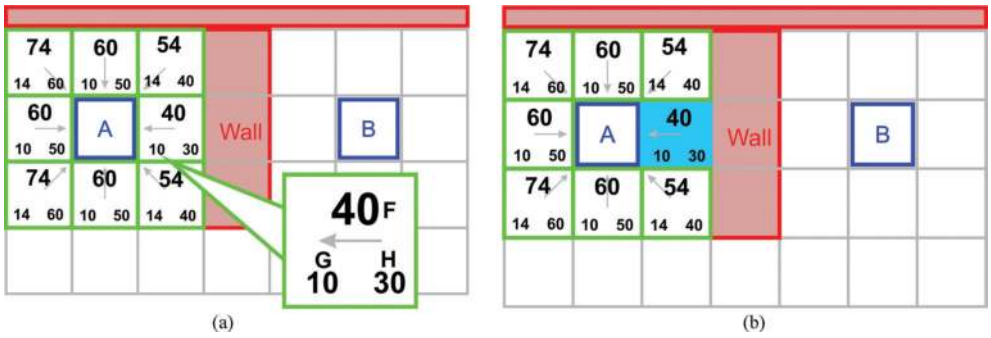

Figure 9. A-star algorithm. (a) First iteration of the algorithm and (b) the lowest cost node.

corner node of A is $H = 40$ because four nodes are required to get there from B. Thus, the lowest value of F is 40 being the right node chosen as the first move that must be performed by the robot (see **Figure 9(b)**). Also, this node is added to the close list to avoid considering it again.

During the next iterations of the algorithm, the new initial position is the node resulting from the last iteration; so all previous nodes in the open list are moved to close list, and the open list will contain the neighbor nodes to such initial node. In addition, the values of $F(x)$, $G(x)$ and $H(x)$ are updated to search for the next low cost node. In our example, only four nodes are added to the open list corresponding to up, down and diagonal nodes to the node under evaluation. Thus, the next node added is the lower right corner node, as it throws the lowest value of $F(x) = 54$.

Once again, the iterative process starts by adding new neighbor nodes to the open list and computes the cost function to find the closest node in the open list with the lowest cost. This recursive process ends when the final node is added to the close list. **Figure 10(a)** shows the nodes added to the close list after five iterations of the A-star algorithm in blue. On the same figure, image b shows the final shortest path possible, starting from the goal B and going to the parent node backwards.

3.2. Programming the palletizer robot

As mentioned before, the NXT brick can be programmed in LabVIEW code to perform the robotic arm movements:

1. Move up and down the arm.
2. Open and close the gripper to take the box.
3. Open and close the gripper to leave the box on the pallet.

Moreover, to perform a safe and coherent navigation, the path planning algorithm is programmed on the sb-RIO reprogrammable card of the robot, that is, following the abovementioned A-star technique. In this chapter, we explain robotic arm motion and pathfinder algorithm.

The LabVIEW Robotics software provides a module of the A-star algorithm which computes an optimal path to get to goal position [18]. The path includes horizontal and vertical trajectories of the robot, left or right rotations and diagonal displacements. However, our mobile robot can only

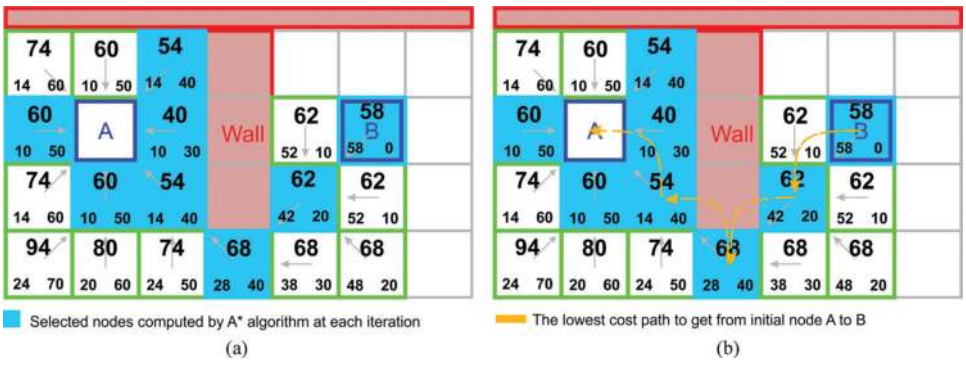

Figure 10. (a) Nodes computed by the A-star algorithm at each iteration and (b) path to get from node A to node B.

perform horizontal and vertical trajectories and left or right rotations. Diagonal movements cannot be performed, because the robot size is not considered for the final path computation, and, consequently, it is more important to avoid possible robot collisions than to get an optimal traversing path. For this reason, the virtual instrument (VI module) provided on the software is modified to exclude all diagonal nodes that could be included in the path. The modified class is called "*GetNeighbours*" of the library "*OccupancyGridWorldMap,"* and only we establish a fixed threshold value for the cost function of the diagonal nodes, such that it will always be higher than other possible nodes. **Figure 11(a)** illustrates simulation results of the A-star algorithm, and **Figure 11(b)** the modified version for the same environment before programming the mobile robot DaNI. Note that there are diagonal displacements on the path, yet they are not too close to the obstacles, so the risk of collision is minimal. Real tests of the final palletizer robot are presented in the next section.

4. Experimental results

A 3D model of the palletizer robot designed on Google SketchUp® is shown in **Figure 12**.

The experimental tests were performed on room with natural light, without any kind of obstacles around. The robot moves on a rectangular wood base sized 2 × 1.2 m, and the pallet is a square wood base of 30 cm located on the upper right corner of the base. Initially, the robot is located at the lower left corner as its start point. This information about the environment is registered as a matrix in a text file, with values '0' and '1' representing free and occupied cells in the environment, respectively. The size of the matrix is related to the navigable space, and in our case, the matrix is 20 × 12; therefore, each cell is 10 cm big, representing an environment of 2 × 1.2 m.

Some images of the robot arriving at the object location and picking it up with the gripper are included in **Figure 13**. Many orders are carried out during this first routine of the robotic arm: set, down, open, close and up gripper to take the object. The second routine of the robotic arm consists in placing the object on the final pallet. This routine starts with the arm up, and then, it goes down slowly. Once the gripper is located over the pallet, it opens, goes up and finally closes the gripper (**Figure 14**).

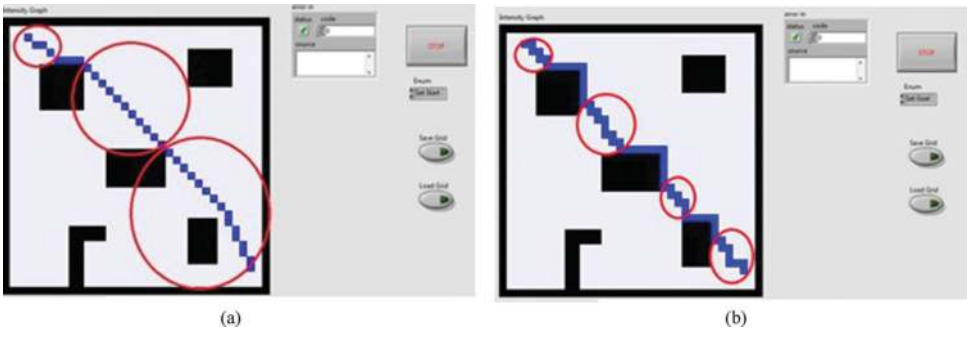

Figure 11. (a) Simulation results of the A-star algorithm and (b) results of modified A-star algorithm.

http://dx.doi.org/10.5772/intechopen.72872

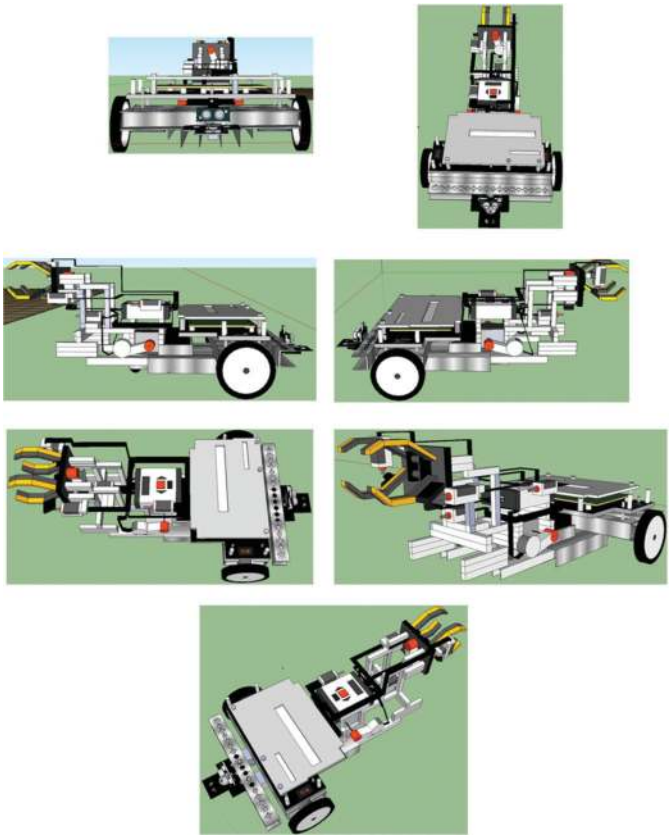

Figure 12. 3D model of the assembled palletizer robot.

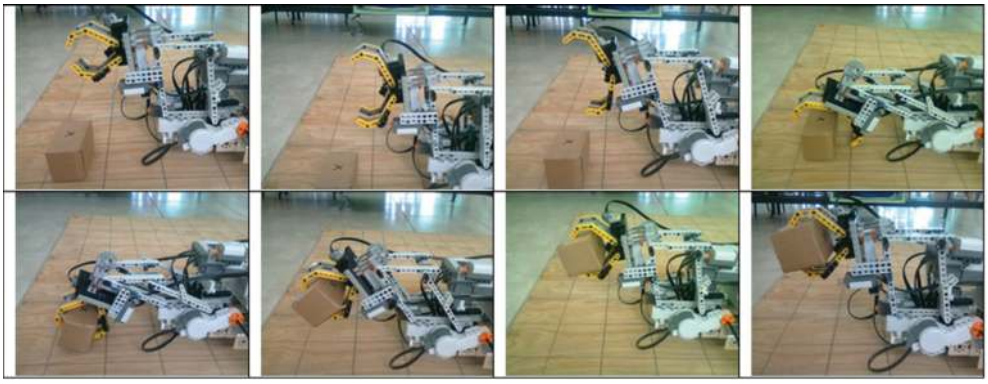

Figure 13. Images illustrate the first routine performed by the robotic arm: picking up an object.

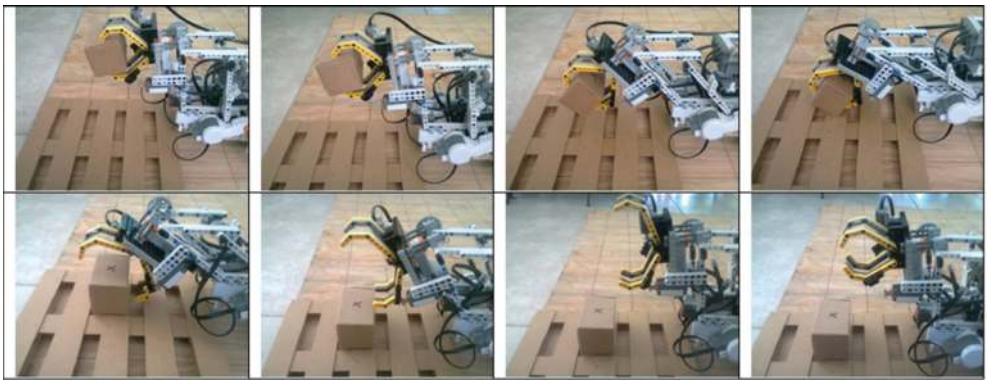

Figure 14. Images illustrate the second routine performed by the robotic arm: placing object on the pallet.

The robotic arm motion is a program synchronized with the path planning strategy in the mobile robot; in contrast, the A-star algorithm was established as the main routine of the palletizer robot. That is, rather DaNI robot or NXT Lego performs their programmed tasks sequentially and the mobile robot has the master control. The path planning module on the robot consists of three stages: initial, intermediate and final. The first stage involves the A-star algorithm: it receives the initial map of the environment (the text file explained above), the start and end positions of the robot, and then, the module calculates the navigable path of the robot. In addition, this stage calculates an occupation map, indicating the cells that the robot must "visit" during navigation to get to the goal position.

In the intermediate stage, the occupation map with the resulted trajectory is analyzed and adequate in accordance with the robot dimensions and motor power: set velocity parameters, rotation angles, and warranty that all the movements could be performed by the DaNI robot. The interpretation and adequacy of the path in the robot consists in setting up when the robot moves straight, turns left or right or when the goal is reached. Thus, a straight motion is included (represented as '0') when the coordinates (x,y) of any pair of consecutive cells in the path change only in the 'x' or 'y' value. If both coordinates change, then a left or right motion ('1') is included depending on the change in the coordinates. As the cells in the path are evaluated in pairs of cells, the robot stops when the second (x,y) cell coordinates is the goal, represented as '2'. In this way, at the end of the intermediate stage, a chain of instructions that includes numbers '0', '1', '2' codes the physical movements that the robot must perform.

During the final stage, the movements and rotations planned for the robot are carried out in accordance with the chain of instructions provided by the intermediate stage. **Figure 15** illustrates a graphical interface that simulates robot navigation in an environment of 36×36 cells with obstacles, for an initial point and final point of $(1,1)$ and $(20,20)$, respectively. The output path is shown in the left graph, while the output data of initial, intermediate and final stages, including the chain of instructions, are illustrated at the top of the graphical interface.

http://dx.doi.org/10.5772/intechopen.72872

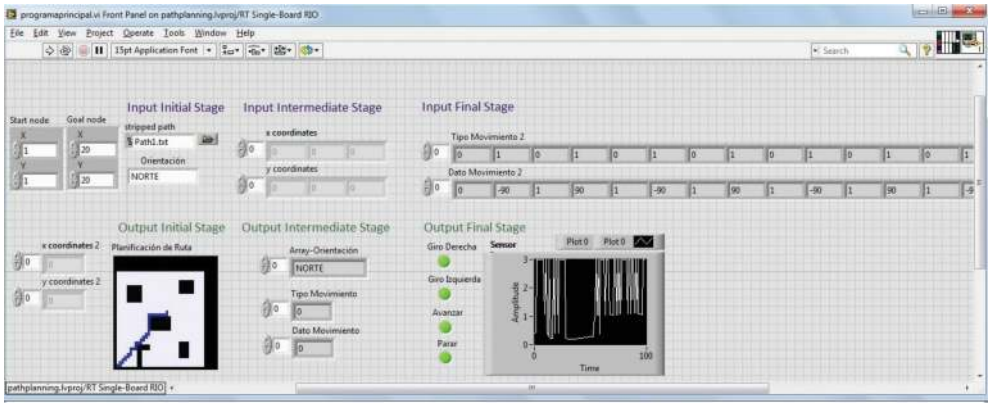

Figure 15. Graphical interface of the robot navigation.

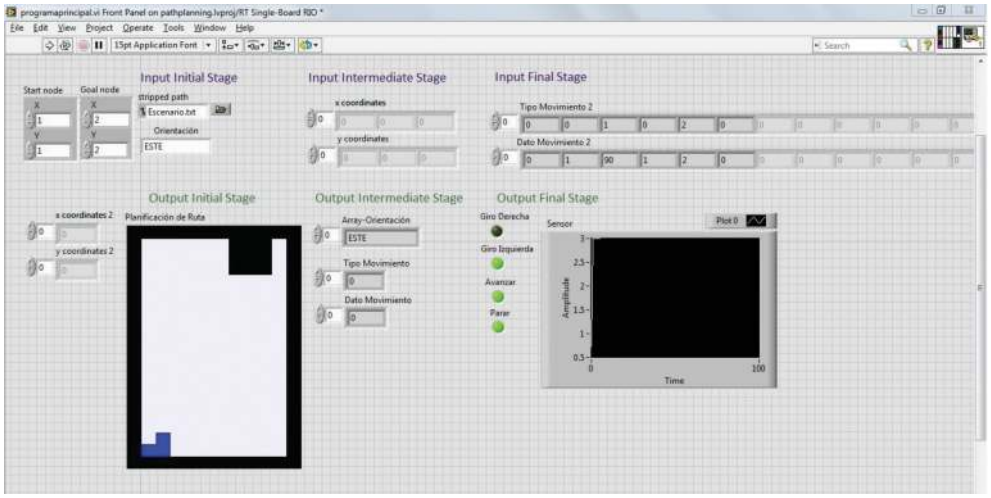

Figure 16. First path to move the robot from (1,1) initial position to (2,2) box location, without obstacles in the environment.

In our real environment, the first test carried out was to move the robot from (1,1) cell (initial robot position) to (2,2) cell (box location), without obstacles on the map (**Figure 16**). **Figure 17** shows the first part of the proposed routine, when the robot takes the box.

The second path performed consists of getting to a second final point, that is, the pallet location for leaving the box. Here the initial position will be the object location which corresponds to the final coordinate performed in the first stage. Thus, considering the pallet location as cell *(8, 3)*, the second path performed to get the pallet is illustrated in **Figures 18** and **19**: graphical interface results and the second part of the real robot performance.

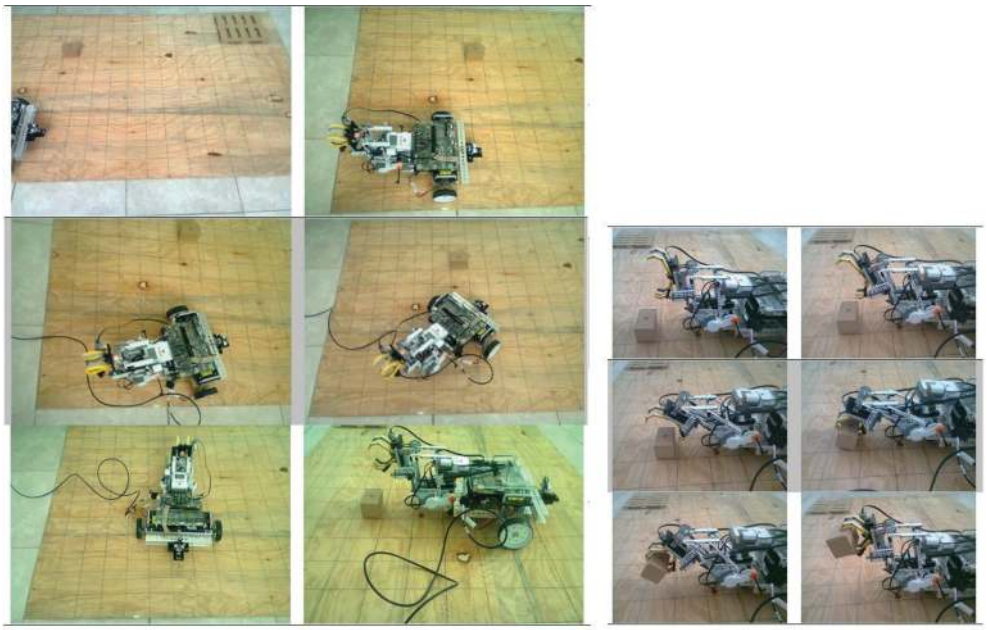

Figure 17. Real robot performance. (a) Robot path to the box location and (b) robot takes the box.

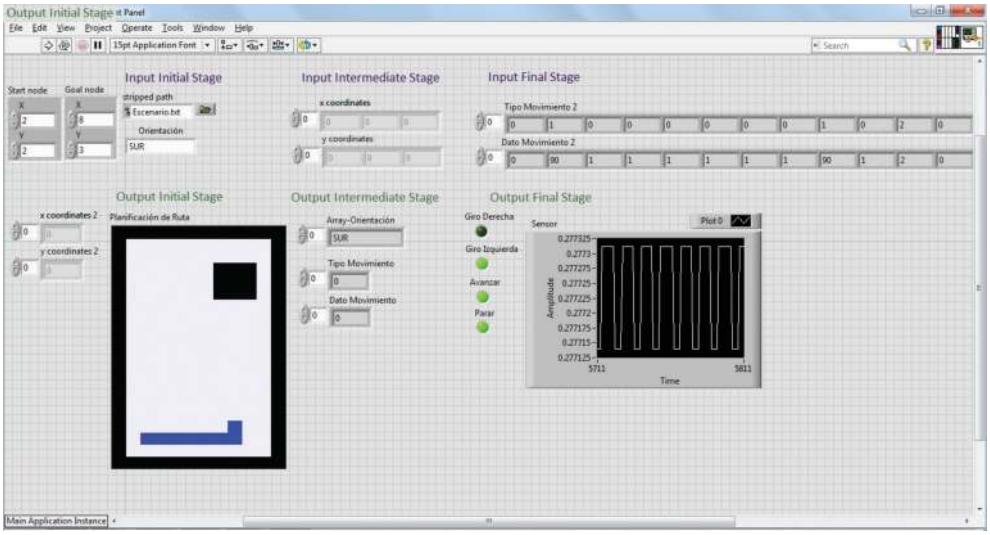

Figure 18. Path to move the robot from (2,2) to (8,3) cells, without obstacles in the environment.

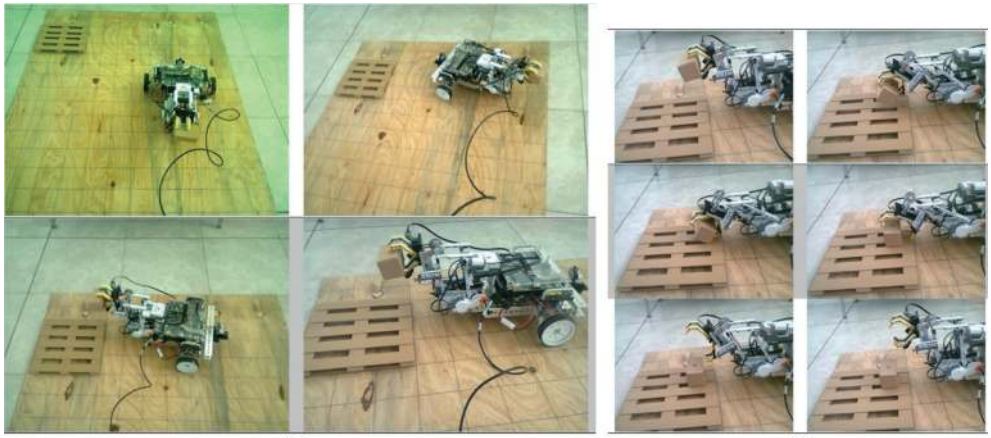

Figure 19. Final part of the routine. (a) By rows: robot moves from initial position to the pallet carry on the box and (b) by rows: robot leaves carefully the box on the pallet.

If no more boxes must be palletized, the robot performs a third path to reach the initial position again, in our example cell (1,1).

5. Conclusions and perspectives

The task of moving an object from one place to another in an autonomous way requires many considerations: the mobile robot, the mechanism used to carry the object, the path planning strategy, and synchronization of all the systems involved in the task. In this chapter, the robotic NXT arm can be programmed at a maximal distance of 10 m. and some of the considerations to implement this kind of project are:

1. Validate ultrasonic sensor values when executing the trajectory.
2. Execute the trajectory in two stages: to reach the box and to place it in the pallet.

An appropriate space must be found for robot navigation purposes to avoid excessive or deficient friction of the wheels against the floor, to establish speeds and distances.

The overall strategy programmed on the mobile robot should encourage beginners or young people interested in robotic developments. We are currently working on the integration of a method to verify if the second phase of the trajectory can be performed as initially established. In addition, we are also correcting odometric mistakes at a mechanical level. On both platforms, DaNI robot and NXT, LabView and LabView robotic modules are used respectively to

program communication between the robotic arm and the mobile robot using the same rules of code. Finally, a perspective to increase the accuracy of the navigation consists in adding a video camera to monitor the palletizing process.

Author details

Dora-Luz Almanza-Ojeda*, Perla-Lizeth Garza-Barron, Carlos Rubin Montoro-Sanjose and Mario-Alberto Ibarra-Manzano

*Address all correspondence to: luzdora@ieee.org

Electronics Engineering Department, DICIS, University of Guanajuato, Salamanca, Guanajuato, Mexico

References

[1] Spong MW, Vidyasagar M. Robot Dynamics and Control. India: Wiley India Pvt. Limited; 2008. https://books.google.com.mx/books?id=PtxYAv7ZUYMC ISBN: 9788126517800

[2] Iocchi L, Ruiz-Del-Solar J, Zant T. Advances in domestic service robots in the real world. Journal of Intelligent and Robotics Systems. 2014;**76**(1):3-4. DOI: 10.1007/s10846-014-0021-1

[3] Zavadskas EK. Automation and Robotics in Construction: International Research and Achievements. Universidad de Alicante; 2012. pp. 2-5. DOI: 10.1016/j.autcon.2009.12.011

[4] Lopez BL. Distribuciones hibridas: Los sistemas de fabricacion flexible [Internet]. Available from: http://www.academia.edu/10375825/DISTRIBUCIONES_HIBRIDAS [Accessed: 02-08-2017]

[5] Almanza-Ojeda DL, Gomar-Vera Y, Ibarra-Manzano MA. Occupancy Map Construction for Indoor Robot Navigation. In: Hurtado EG, editor. Robot Control. Croatia: Intech; ch. 4. pp. 69-87. ISBN: 978-953-51-2684-3, September 2016. http://www.intechopen.com

[6] "VexRobotics" [Internet]. Available from: http://www.vexrobotics.com.mx/ [Accessed: 09-08-2017]

[7] Zowi robot in Europe [Internet]. Available from: http://zowi.bq.com/fr/ [Accessed: 09-08-2017]

[8] National Instruments Corporation. Introduction to the LabVIEW Plataform [Internet]. Available from: http://www.ni.com/webcast/439/es/ [Accessed: 29-09-2017]

[9] Hopkins B, Antony R. Bluetooth for JAVA. USA: Apress; 2003. ISBN: 978-1-59059-078-2

[10] National Instruments Corporation. NI Single-Board RIO Embedded Control and Acquisition [Internet]. October 2012. pp. 1-3. Available from: ftp://ftp.ni.com/pub/branches/northern_region/fpga_kit_feb14/what_is_ni_singleboardrio.pdf [Accessed: 29-09-2017]

[11] Rico JM. Analisis dinamico de un mecanismo plano de cuatro barras. Analisys of Mechanisms, undergraduate course. Division de Ingenierias Campus Irapuato-Salamanca, Universidad de Guanajuato; Mexico. 2014. pp. 2-6

[12] Lent D. Analysis and Design of Mechanisms. USA: Prentice Hall; 1970. ISBN: 978-0-13032-797-0

[13] SolidWorks Corporation. Student's Guide to Learning SolidWorks® Software. France: D'assault Systemes SolidWorks Corporation, PMS0119-ENG; 2011. Available from: https://www.solidworks.com/sw/docs/Student_WB_2011_ENG.pdf [Accessed: 23-06-2017]

[14] Lego Mindstorm NXT Datasheet. Lego Mindstorm Education; Denmark. 2008. 66 p. Available from: https://www.generationrobots.com/media/Lego-Mindstorms-NXT-Education-Kit.pdf

[15] Trimble: How to use Google-sketchup, Trimble Corporate and SketchUp developers. Available from: https://www.sketchup.com/learn/videos/826 [Accessed: 14-08-2017]

[16] Dechter R, Judea P. Generalized best-first search strategies and the optimality of A*. Journal of the ACM. 1985;**32**(3):505-536. DOI: 10.1145/3828.3830

[17] Ferguson D, Likhachev M, Stentz A. A guide to heuristic-based path planning. Proceedings of ICAPS Workshop on Planning under Uncertainty for Autonomous Systems; 2005. www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/maxim/files/hsplanguide_icaps05ws.pdf

[18] Gretlein S. Presentando labview robotics: de la fantasia a la realidad. Instrumentation Newsletter. 2010;**22**(2):3-5. Available from: ftp://ftp.ni.com/pub/gdc/tut/abril-junio_2010.pdf