The analysis of the robotized handling of oxygen gas bottles with the help of the Petri networks

Lavinia Culda[1], *Elena Stela* Muncuț[1*], and *Geza* Mihai Erdodi[1]

[1]University "Aurel Vlaicu" of Arad, AIITT Department, B-dul Revoluției, nr.77, 310130, Romania

Abstract: The gas bottling industry is facing a number of problems. In this paper we accomplish a study on the robotic storage of oxygen gas tanks (bottles) applicable to other gases too. To do this, we developed a prototype using the Lego Mindstorms kits and we simulated the process to study the efficiency with the Petri Networks. We chose this kit, due to its complexity, being the closest to what we needed to make small scale storage. The program the robot uses for working was done in a special programming language: Mindstorms SDK and downloaded to the controller's memory.

1 Introduction

To study the communication with programmable machines, Carl A. Petri created a mathematical networking tool in 1962 that highlighted the process synchronization, the conflict resolution, or the resource sharing. [6-7] These properties characterize the discrete event systems including automated industrial systems, communication systems and computer-based systems and transform the Petri networks into a promising tool and a technology for applications in industrial automation.

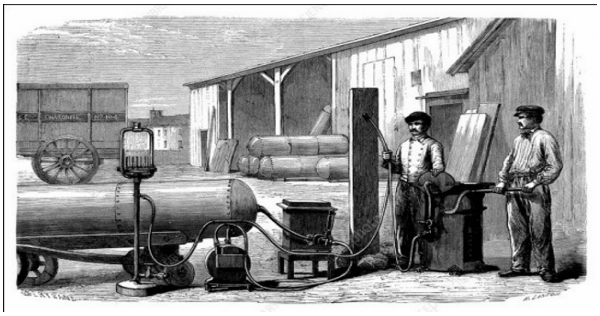


Fig. 1 Gas cylinders, 19th century. Illustration of workers handling compressed coal gas in cylinders, using a system devised by Lapparent. [1-3]

For the case study, we opted for a less publicized industry, namely the Oxygen Industry. Studying a local oxygen bottling plant, we found that: the working environment has a high degree of danger, unprofitable and outdated flow of bottling, the storage is inefficient and the number of workers used to carry out simple operations is quite high.

* Corresponding author: muncutstela@yahoo.com

The aim of the paper is to find modern solutions for: increasing work safety, decreasing the number of workers, promptly serving the customer and make the flow more efficient using the Petri mathematical modeling.

A Petri net, also known as a place/transition (PT) net, is one of several mathematical modeling languages for the description of distributed systems. It is a class of discrete event dynamic system. A Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e. events that may occur, represented by bars) and places (i.e. conditions, represented by circles). The directed arcs describe which places are pre- and/or post conditions for which transitions (represented by arrows). [17]

Like the industry standards such as UML activity diagrams, Business Process Model and Notation and EPCs, Petri nets offer a graphical notation for stepwise processes that include choice, iteration, and concurrent execution. Unlike these standards, Petri nets have an exact mathematical definition of their execution semantics, with a well-developed mathematical theory for process analysis. [17]

A Petri net consists of places, transitions, and arcs. The arcs run from a place to a transition or vice versa, never between places or between transitions. The places from which an arc runs to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition. [17]

Graphically, places in a Petri net may contain a discrete number of marks called tokens. Any distribution of tokens over the places will represent a configuration of the net called: marking. In an abstract sense relating to a Petri net diagram, a transition of a Petri net may fire if it is enabled, i.e. there are sufficient tokens in all of its input places; when the transition fires, it consumes the required input tokens, and creates tokens in its output places. A firing is atomic, i.e. a single non-interruptible step. [17]

Unless an execution policy is defined, the execution of Petri nets is nondeterministic: when multiple transitions are enabled at the same time, they will fire in any order. Since firing is nondeterministic, and multiple tokens may be present anywhere in the net (even in the same place), Petri nets are well suited for modeling the concurrent behavior of distributed systems. [17]

The networks contain two disjoint sets of elements: state elements, representing passive objects of the modeled system, and transition elements for modeling the active objects.

Petri Networks are a special category of graphs. A graph is completely defined if the sets of nodes and arcs are known. The difference between a graph and a Petri network is that in the latter case the set of nodes is replaced with two disjoint sets:

- The set of places Pi, i = 1, ..., n (represented by circles)
- The set of transitions Tj, j = 1... m (represented by vertical bars or squares).

The arcs of a Petri network are unidirectional. An arc can only bind a transition from a place or a place of transition. At a transition or a place there can be more arches, and from a transition or a place can also go more arches.

The matrix that contains the evaluations of the arcs of a Petri network is called the incidence matrix. The element on line i and column j of incidence matrix A has the value of the arch valuation connecting the node Pi to the transition Tj if Tj is an input transition at the Pi node.

If the transition Tj is an exit transition from Pi, then the corresponding element of the matrix A has the same value of the corresponding arch valuation, but with the changed sign.

If there is no arc between the node Pi and the transition Tj, then the corresponding matrix of incidence element is null.

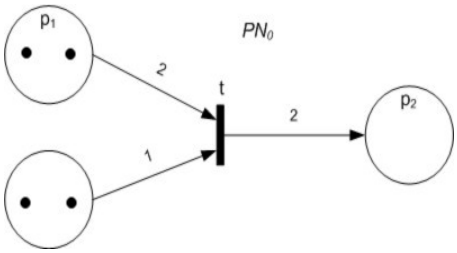


Fig. 2 A Petri net with an enabled transition [17]

The Petri net that follows after the transition fires (Initial Petri net in the figure above) [17]. Marking a Petri Network means an application that associates each place in the network with an integer represented by as many points (tokens) within the circle that symbolizes that place. Not any Petri network needs to have a tag. Those that have attached that application are called Petri marked networks. [17]

The markings of a Petri net (S, T, W, M_0) can be regarded as vectors of nonnegative integers of length |S|. [17]

Its transition relation can be described as a pair of |S| by |T| matrices.

$$W^-, \text{defined by } s,t : W^-[s,t] = W(s,t) \quad (1)$$

$$W^+, \text{defined by } s,t : W^+[s,t] = W(t,s) \quad (2)$$

Then their difference

$$W^T = W^+ - W^- \quad (3)$$

It can be used to describe the reachable markings in terms of matrix multiplication, as follows. For any sequence of transitions *w*, write o (w) for the vector that maps every transition to its number of occurrences in *w*. Then, we have

$$R(N) = \{M | \exists \omega : M = M_0 + W^T \cdot o(\omega),\ \omega \ is\ a\ firing\ sequence\ of\ N\} \quad (4)$$

Note that it must be required that *w* is a firing sequence; allowing arbitrary sequences of transitions will generally produce a larger set. [17]

2 Description of the oxygen bottling flow

At present the transport of gas-containing bottles (gas tanks) is rudimentary by means of trolleys, and the storage is done in enclosures. In modern factories these things are automated.

To improve the efficiency of the manufacturing process in the oxygen plant, we went to the actual manufacturing flow.

Starting from the problems that have arisen, we have traced a new technological flow consisting of conveyor belts and manipulating robots to limit the access of the human factor to the problematic areas and we designed all the auxiliary sections in the work area to minimize the dead times.

Here's how this flow works (figure 3). The empty bottle arrives at the ramp and is taken over by the bottle receiving point. Each cylinder's data is entered into the computer and receives a bar code for identification. After that, the bottle is placed on the conveyor belt. If the bottle is good, it will go to the empty bottle depot and if it is defective (it has visible defects or the date of the last technical check is passed) it is sent to the defective bottle depot, where it is subsequently removed from the factory and sent for repair.

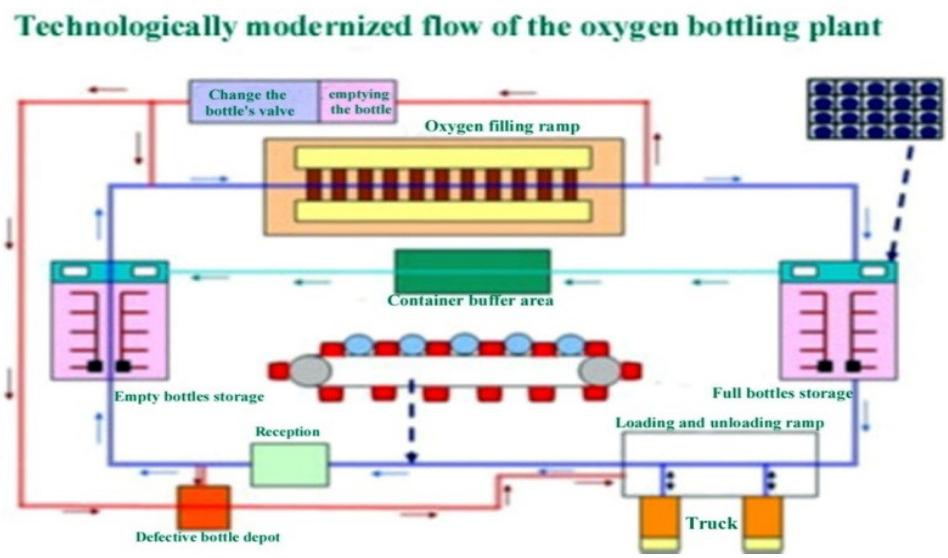


Fig. 3 Technological flow of oxygen bottling

In the case of good bottles, the conveyor belt moves the bottle inserting it into the empty bottle depot. In the storage it is taken over by the storage robot (DAH), which brings it into the free storage location for storage. The position sensors send to the central computer the cylinder's position coordinates. This process is applied to all cylinders.

When the bottles are removed for filling, the central computer gives the command to remove bottles of the same type. They are placed in a container that once loaded, is carried by the conveyor belt to the filler. The number of containers varies depending on the capacity of the plant and necessity. They do not leave the factory. They only have the role of supporting the bottles on filling, so they are recirculated within the flow. When they are not in the stream, they will be stored in a buffer depot.

Once the bottles have been charged with oxygen, they are transported to the full bottle depot. If it is found during the filling that one of the bottles is defective, then, at the exit of the filling container, the defective bottle is removed and transported by means of the conveyor belt to the bottle emptying ramp. The capacity of the ramp is 2 bottles. Here the elimination of the amount of oxygen present in the cylinder takes place. After completion of emptying, the bottle is repaired. The only possible repair in the factory is to change the cylinder's valve. If this is not possible or there are other failures, the bottle is sent to the defective bottle depot. If the repair is possible, the bottle returns and is recharged.

The bottle container reaches the full bottle depot where the container is emptied and the bottle is placed in the rack until the recipient arrives. The empty container is transported to the buffer area where will wait for the empty bottle charging request.

At the arrival of the beneficiary, the removal order of the cylinder is sent to the DAH (storage robot), and the bottle is taken to the truck (camion) at the loading ramp by the conveyor belt.

The empty bottle depot is designed as storage with metallic shelves provided with a finite number of places where the bottles can be introduced.

The bottles are stored horizontally and thus the number of bottles stored is much higher than if the bottles were stored in the upright position.

The shelves are distributed over several rows and have multiple levels. The distance between them is large enough to allow DAH to move in good condition. DAH moves on a well-defined route and has the role of taking the oxygen bottle from the conveyor belt and

depositing it in the storage, respectively, removing it from the deposit and depositing it on the conveyor belt.

The bottle moves with the conveyor from one point of the technological flow to another. The belt is provided with transverse ribs made of rubber to avoid collision of bottles during transport.

In the case of full bottle storage, the storage procedure is the same but upon removal from the storage, DAH receives the order to remove the bottle requested by the customer from the storage. This is placed on the conveyor and handed over to the loading / unloading ramp from where it is taken over by the customer.

3 Technical flow simulation with the lego mindstorm rcx kit

In this paper we are conducting a study on the robotic storage of oxygen bottles. For this we made a prototype using the LegoMindstorm kits. Buying a "real" robot that works in robotic factories is very costly, while purchasing of such a kit is available to anyone.

We chose this kit to design the machete, due to its complexity, being the closest to what we needed to make a small-scale warehouse.

The model we made (figure 5) has as constituents the elements existing in the warehouse, no matter if they are full or empty cylinders, namely: the conveyor belt, the robot and the rack.

The conveyor belt is made of a stainless-steel profile with a length of 70 cm and a width of 10 cm. At the ends are two wooden cylinders that are intended to move the textile tape, with cross-separators to support the oxygen bottles during the transport. The movement is generated by a 9-volt motor that transmits the roller movement by means of a gear tooth drive. The motor is powered by the battery, but can also be powered from another power source.

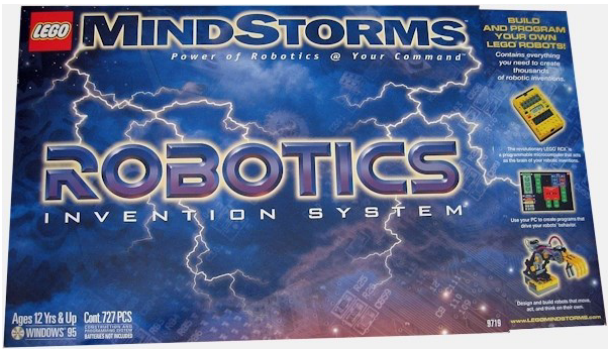


Fig. 4 LegoMindstorm rcx kit, [4]

An optical sensor is attached to the conveyor belt. The sensor detects the proximity of the bottle to the sensor. This sensor senses the change of the luminous flux in the presence of the bottle to be gripped. The proximity of the object interrupts the light beam, causing a signal to be transmitted to the controller. Once the signal has been received, the conveyor belt stops and the robot start to move. If the light beam is not interrupted, the belt keeps being in motion.

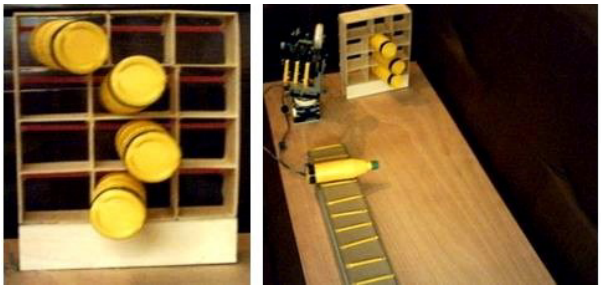


Fig. 5 An optical sensor

The rack is designed to store the bottles in horizontal position. Because we used a static robot to make the model, the positions occupied by the bottles are those corresponding to the trajectory described by the robot's arm. The shelf positioning is based on an initial schedule that takes into account the trajectory described by the robot's arm. Such a storage system is safer and more efficient than the classic system in a horizontal position in the pens.

The warehouse robot has the role of retrieving the bottles from the conveyor belt and introducing them into the rack. The robot is completely independent and is commanded by a controller.

The controller has the ability to "track" 3 sensors and 3 motors. On the robot there are 3 engines that generate the movement that is transmitted through gearwheel and wheel belt. We've linked two touch sensors and an optical sensor to this controller. The optical sensor is the one used by the conveyor belt and the two touch sensors are fixed on the robot.

Fig. 6 The construction of the robot

The robot's working program was made in a special programming language: Mindstorms SDK and downloaded to the controller's memory by a tower, using infrared transmission.

4 Analysis with the petri network

On the basis of the modernized flow, we carried out a simulation using the Petri Networks to study the efficiency of this system.

For the simulation the FIFO prioritization system has been used. For each workstation was assigned a P0-Pn position represented by a circle, and to the transformation process was assigned a T0-Tn transition. [5] The arcs are the link between positions and transitions. The bottles are marked with black spheres called chips that will go through the simulated positions and transitions. The HPSim simulation program [8-16] gives us the opportunity to see in real-time the blockages that may appear on the traced network. The places in a Petri network serve, among other things, to model transport elements within the same process.

Being a closed area, it was impossible to get real times, and based on the information from the engineers at the factory, we outlined more theoretical times but they could also be encountered in a real modern stream. The simulation was done using HPSim software.

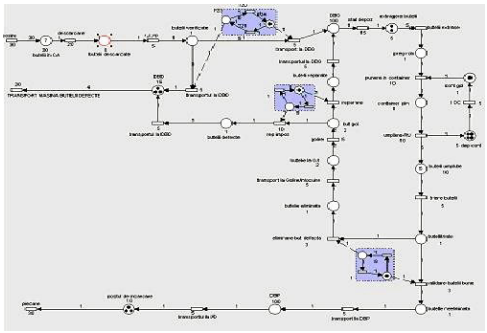


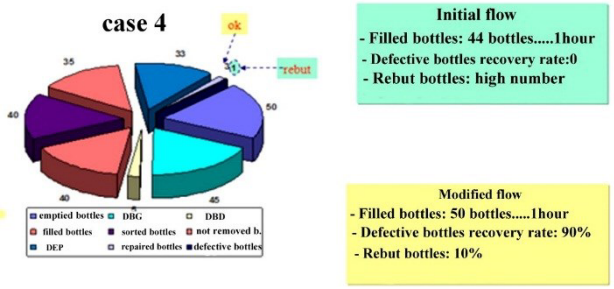
Fig. 7 Model graph

Using the network from Figure 7, we've done several tests to see how positions are loaded based on the time variation of the simulation. Resulting of the tests performed for different time and step values we obtained the values from the Table 1 below. The number of bottles tested was 50 and the result can be seen in Figure 8.

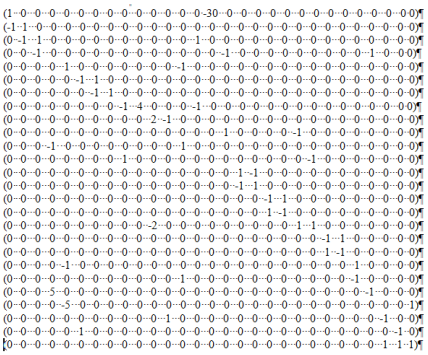
We used for modeling: 26 positions and 28 transitions

Table 1 . Resulting of the tests

Selected time (ms)	**10.000**	**1.000**	**500**	**1.000**	**1.000**
The number of steps	**10.000**	**10.000**	**796**	**1677**	**1.000**
Unloaded bottles	50	50	24	50	**29**
Checked bottles	50	50	24	50	**29**
Empty bottles storage	49	50	21	45	**26**
Bottle extraction	49	50	15	45	**26**
Full containers	9	10	3	8	**5**
Filled bottles	45	50	15	40	**20**
Sorted bottles	45	50	15	40	**20**
Removed bottles	6	9	0	5	**0**
Not removed bottles	39	41	15	35	**20**
Emptied bottles	6	9	0	4	**0**
Repaired bottles	5	8	0	3	**0**
Defective bottles	1	1	0	1	**0**
Bad bottle storage	7	4	3	5	**3**
Removing flawed bottles from the factory	4	4	0	5	**0**
Full bottles storage	39	41	15	33	**20**
Truck loading station	**39**	**36**	**10**	**32**	**20**



a) Incidence Matrix:



b) Marking Vector: (7 0 0 0 0 0 3 3 0 0 0 0 0 0 1 0 1 0 1 0 4 1 2 5 0 0 0)

Fig. 8 The result of the simulation

The advantages of this solution would be:
- the number of human operators (workers) in the factory decreases;
- the number of accidents due to human error decreases;
- quality of service is growing;
- this is a multi-purpose system: transport, storage;
- it can be easily applied to other gas-containing bottles;
- the container size is variable depending on the size of the bottle we use;
- the cost of implementing the system is low and the investment is amortized in a short time.

5 Conclusion

By modeling and simulating this system, it was desired to highlight the modeling power of timed stochastic colored Petri networks. This is the concrete way of studying as complex as possible real systems.

Physical modeling of the system and then simulation with Petri Networks contributes to the understanding and improvement of the gas bottling system,

Although this system seems to be simplistic, it is good to try to build a more realistic model with the inclusion of the information that can influence its proper functioning. This is achieved both by defining the boundaries of the analyzed system and by defining simplifying assumptions. If you attempt to model all factors and relationships, the model could become too complicated to solve. Therefore, it is necessary to make a compromise between the need to build a simple and easy to solve model and the need to obtain by model a reasonable and plausible representation of the real problem

References

1. https://www.sciencephoto.com/media/631675/view
2. http://www.ct-assocs.com/img/gasgrab_ex2.jpg
3. http://gasbottleyoireka.blogspot.com/2017/02/gas-bottle-safety.html
4. http://www.mindstormsrobots.com/lego-mindstorms/using-lego-mindstorms-ris-rcx-kit-with-windows7
5. http://www.aut.upt.ro/~loredanau/teaching/LIC/Retele%20Petri.pdf
6. http://webdiis.unizar.es/~cmahulea/papers/Pastravanu_Aplicatii_retele_Petri.pdf
7. https://web.archive.org/web/20080911020423/http:/www.airproducts.com/NR/rdonlyres/7CCE748B-35BA-45B2-93D4-48AE3D3EEDF3/0/reference_cylinder_information.pdf
8. http://www.winpesim.de/default.html
9. E. Gutuleac, MEPSCRP, (EDP al UTM, Chisinau 1997)
10. K. Jensen, CPNs, **1,2,3** (Springer-Verlag Berlin Heidelberg, 1992)
11. K. Jensen, L. M. Kristensen, *Coloured Petri Nets-Modelling and Validation of Concurrent Systems*, (Springer-Verlag Berlin Heidelberg, 2009)
12. G. Belgiu, „*Curs Bazele proceselor de fabricație*", UPT, (2008)
13. I. Boros, A. Gherman, H. E. Babeu, AGIR **nr. 1-2** (2008)
14. I. Bondrea, „Modelarea şi simularea proceselor de producţie", (Ed. Universităţii din Sibiu, 1998)
15. A. Ch. Chung, „*Simulation, Modeling. Handbook. A Practical Approach*", (Industrial and manufacturing engineering series. CRC Press LLC, ISBN 0-8493-1241-8, BR London, New York Washington, 2004)
16. L. Culda, PMURI , (Editura Universității Aurel Vlaicu Arad,2018)
17. https://en.wikipedia.org/wiki/Petri_net