U.P.B. Sci. Bull., Series C, Vol. 73, Iss. 3, 2011 ISSN 1454-234x

MULTITASKING CONTROL OF VISUALLY GUIDED ROBOT FOR INTELLIGENT PART FEEDING

Mihai PÂRLEA[1]

Această lucrarea prezintă un post de lucru robotizat pentru realimentarea cu piese în vrac ce conține doua buncăre automate. Sunt tratate problemele de comunicație cu buncărele, prinderea bazată pe vedere a pieselor şi controlul multitasking. De asemenea sunt prezentate date de performanță pentru compararea folosirii unui singur buncăr față de ambele buncăre.

This paper presents a robotic bulk parts supplying workstation which features two part feeders. Feeder communication, visual parts grasping and multitasking control problems are treated. Also, single feeder versus dual feeders performance data is presented.

Keywords: resupply workstation, bulk parts feeder

1. Introduction

The present economy trends demand that production cells equipped with robotized workstations offer the best possible performance and behave predictable over time. Whether you are using a hierarchical [1] or a heterarhical [2] control system, the depletion of a workstation's supply of a certain type of parts will, at least momentarily, interrupt the production activity [7, 8]. So, the conclusion is that we need to include the parts resupplying process in the overall production schedule.

Another problem with the parts resupply process is that the parts arrive in bulk and need to be manually arranged in the workstations storage matrix [9]. This is a repetitive operator task that should be automated.

The solution to both problems is simple: a dedicated parts resupplying workstation connected to the production cells closed loop conveyor.

2. Vibrating part feeders

Bulk part feeders have up until now been included in the workstation that needs that certain type of part [3]. In a dedicated bulk parts resupply workstation, the parts feeder has to scatter a small number of parts on a flat surface, this surface

[1] PhD Student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, e-mail: mihai.parlea@cimr.pub.ro

is inspected by a camera which recognizes usable parts and commands the robot to pick and place them on the conveyor (or on a pallet moved by the conveyor).

Normal operation can be presented as two different phases: feeder prepares parts - robot pick parts. These two phases are repeated until the desired number of picked parts is reached (or the workstation is commanded to stop picking parts). Long term usage of this kind of workstation has shown that the two phases take roughly the same amount of time to execute, so the workstation is placing parts on the conveyor for only 50% of his operating time.

In trying to improve this performance indicator, we added a second part feeder in the workstation and tried to get the most benefit out of this idea. The workstation setup is presented in Fig. 1.

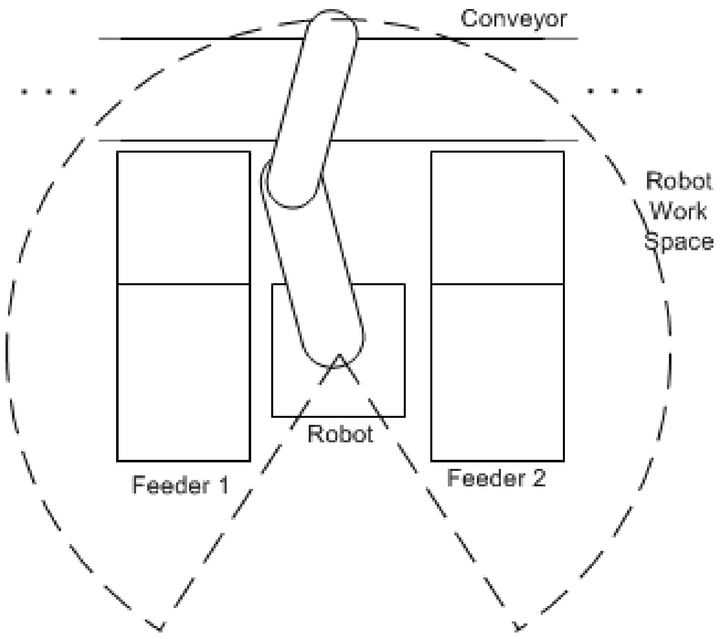


Fig. 1. Dual feeder robotic workstation

The workstation features a horizontally articulated SCARA type robot. Both parts feeders and a portion of the conveyor are placed inside the robot's working space.

This feeder is based on two vibrating surfaces: Bulk Container and Feed Surface. Its role is to chose a couple of parts from the main parts storage (which is contained in the Bulk Container) and spread them out on the Backlit Area (of the Feed Surface) in order for them to be inspected by a camera (see Fig. 2 for feeder parts and functioning principle). The camera can be a fixed down-looking or a mobile arm-mounted / hand-held camera, in our case a fixed down-looking camera was used. After the camera inspects the Backlit Area (and possibly command the robot to pick some parts), a decision is taken in order to bring more parts from the Bulk Container or to change the way the parts already present on the Feed Surface are presented to the camera.

In order to achieve this, the feeder can execute several vibrating movements:

- □ Dispense - vibrates Bulk Container and Feed Surface together in order to send parts falling on to the Feed Surface
- □ Feed Forward - moves parts forward on the Feed Surface
- □ Feed Backward - moves parts backward on the Feed Surface
- □ Flip - flips parts on the other side
- □ Flip Forward / Flip Backwards - combines the flip motion with the move forward / backwards motion
- □ Purge - purges parts from Feed Surface and Bulk Container (parts from Feed Surface fall outside the back of the feeder, parts from Bulk Container fall only if the Retainer Gate is manually opened)

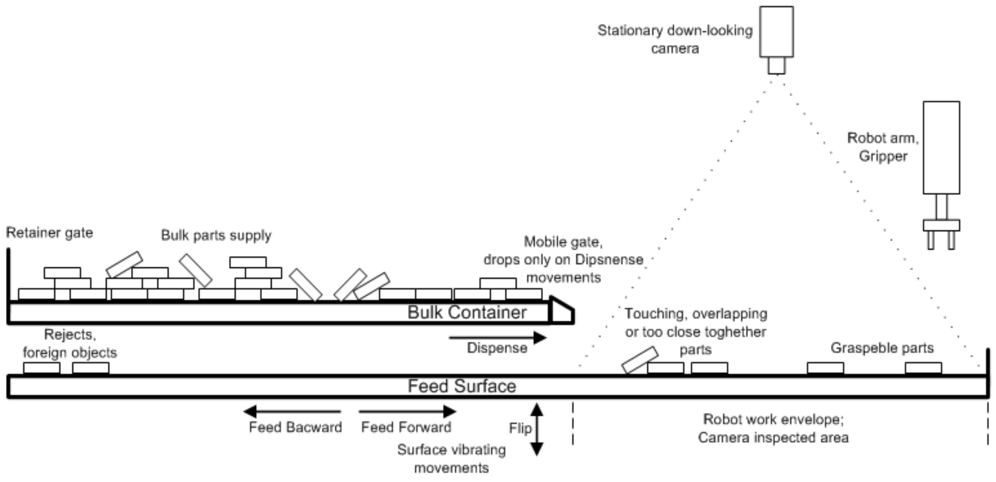

Stationary down-looking camera
Robot arm, Gripper
Retainer gate
Bulk parts supply
Mobile gate, drops only on Dipsnense movements
Bulk Container
Rejects, foreign objects
Dispense
Touching, overlapping or too close toghether parts
Graspeble parts
Feed Surface
Feed Bacward
Feed Forward
Flip
Surface vibrating movements
Robot work envelope; Camera inspected area

Fig. 2. Vibrating parts feeder's surfaces and working principle

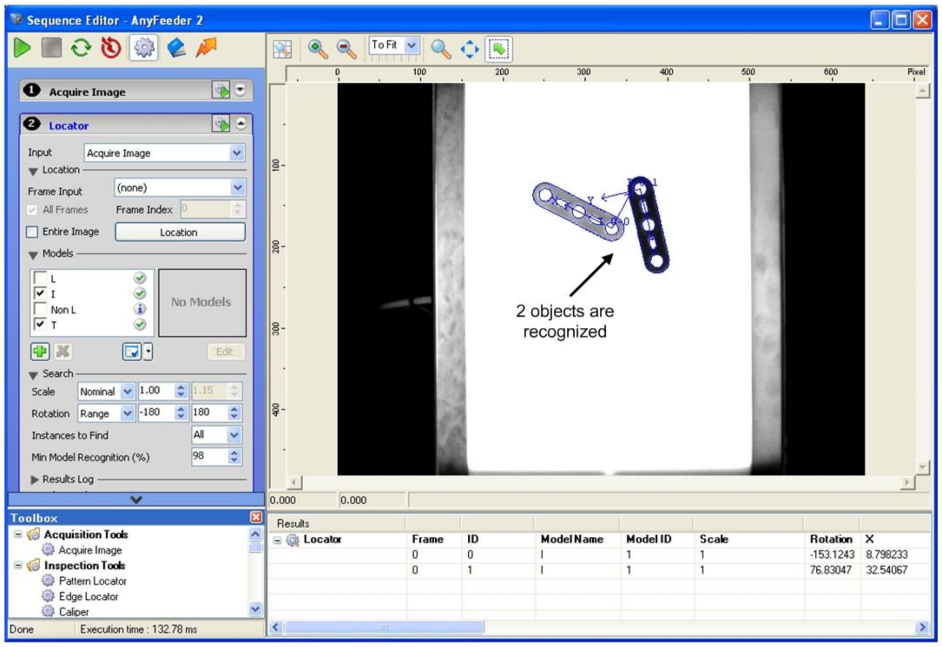
The feeders implement 4 types of commands:

1. initialization
2. movement (already presented)
3. speed
4. turns

3. Vision controlled part recognition and grasping

While working with the visual guidance software, two problems appeared.

The first was resulted from the fact that our robot uses a gripper with parallel fingers; in order to approach the object, this gripper needs to have two free fingerprints next to the object in the grasping position [5]. If a part, recognized by the vision's software locator tool (see Fig. 3), is present on the feed surface on such a manner that an obstacle (such as another part or the feeder's sides) occupies (even partially) one of the gripper's fingerprints, then the gripper will come into collision with this obstacle, resulting in damage tot the robot and to the feeder. In order to detect this situation, I used two Histogram Tools, they are attached to the objects model, each having the gripper's fingerprint dimensions and placed in the grasping positions. For each of these tools the "minimum grey level value" is inspected. If the value is 255, then all the pixels inside the tool are white and the fingerprint is declared free of obstacles, if the value is smaller than 255, it is considered that an obstacle is present and the fingerprint is not declared free of obstacles (see Fig. 4). Only if both the fingerprints are declared free then the object is considered graspable and the robot is commanded to pick it up.



Input: Acquire Image
Location
Frame Input: (none)
All Frames ✓ Frame Index: 0
Entire Image ☐
Models: ☐ L, ☑ I, ☐ Non L, ☑ T
Search
Scale: Nominal 1.00 1.15
Rotation: Range -180 180
Instances to Find: All
Min Model Recognition (%): 98

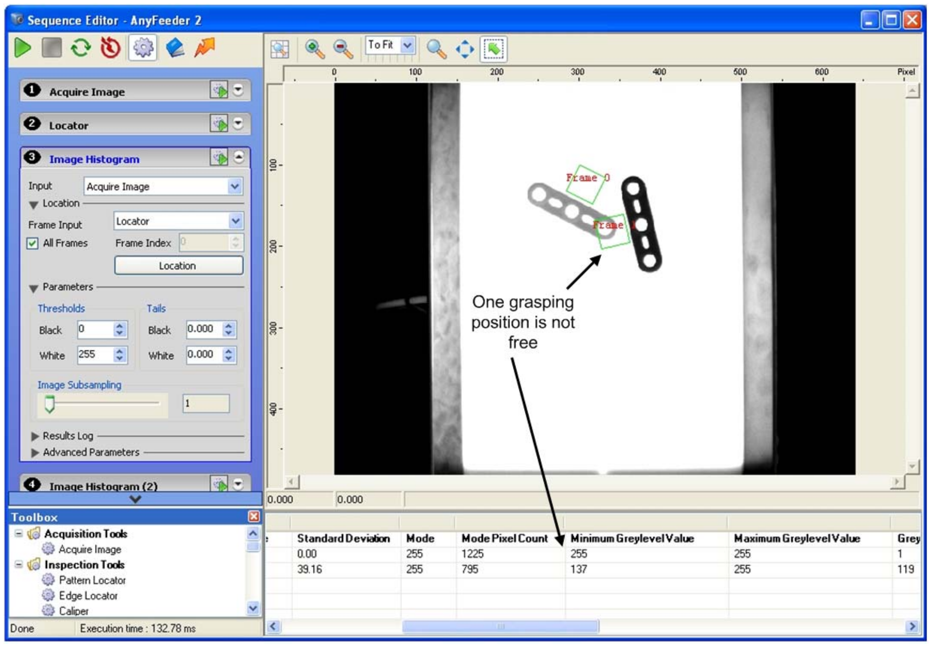
0.000 0.000

Results Locator	Frame	ID	Model Name	Model ID	Scale	Rotation	X
	0	0	I	1	1	-153.1243	8.798233
	0	1	I	1	1	76.83047	32.54067

Done Execution time : 132.78 ms

Fig. 3. Locator detects recognizes two objects

The second problem is that the Locator Tool doesn't recognize overlapping objects (see Fig. 5). This is a good thing, because this robot's gripper cannot grasp overlapping objects and it is difficult to detect which object is on top. But the system needs to know if there are unrecognized objects on the Feed Surface, in order to present them in a different manner to the camera, otherwise, if the system only picks recognized objects and commands the feeder to bring new objects on the Feed Surface, the Feed Surface will soon become clogged with objects that, even if they are recognized, they are not graspable. In order to detect this problem, I used a Blob Analyzer tool. Blobs are areas of adjacent pixels whose gray level satisfy a condition. Since the grey level of free pixels is 255, it means that the pixels occupied by parts have a grey level smaller than 255, so the Blob Analyzer is set up to detect blob's that have a smaller than 255 grey level (see Fig. 6).



Input: Acquire Image
Location
Frame Input: Locator
[✓] All Frames
Frame Index: 0
Parameters
Thresholds: Black 0, White 255
Tails: Black 0.000, White 0.000
Image Subsampling: 1
Results Log
Advanced Parameters

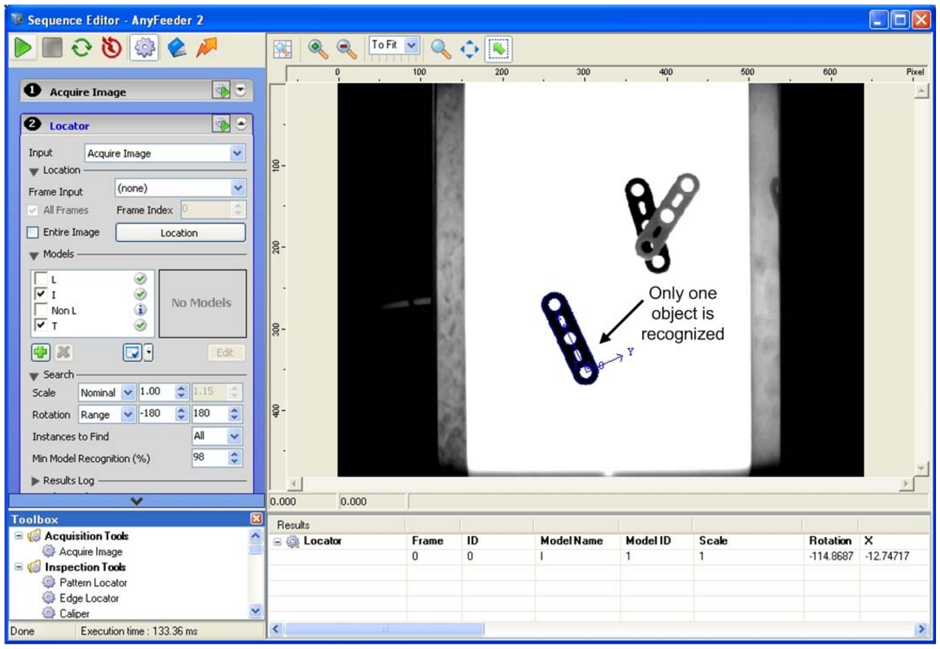
0.000 0.000

Standard Deviation	Mode	Mode Pixel Count	Minimum Greylevel Value	Maximum Greylevel Value	Grey
0.00	255	1225	255	255	1
39.16	255	795	137	255	119

Done Execution time : 132.78 ms

Fig. 4. One object's grasping position is blocked

Then, by comparing the number of recognized objects (1 in this case) with the number of detected blobs (2 in this case), we can determine if the Feed surface contains unrecognized objects (which is true in this case).

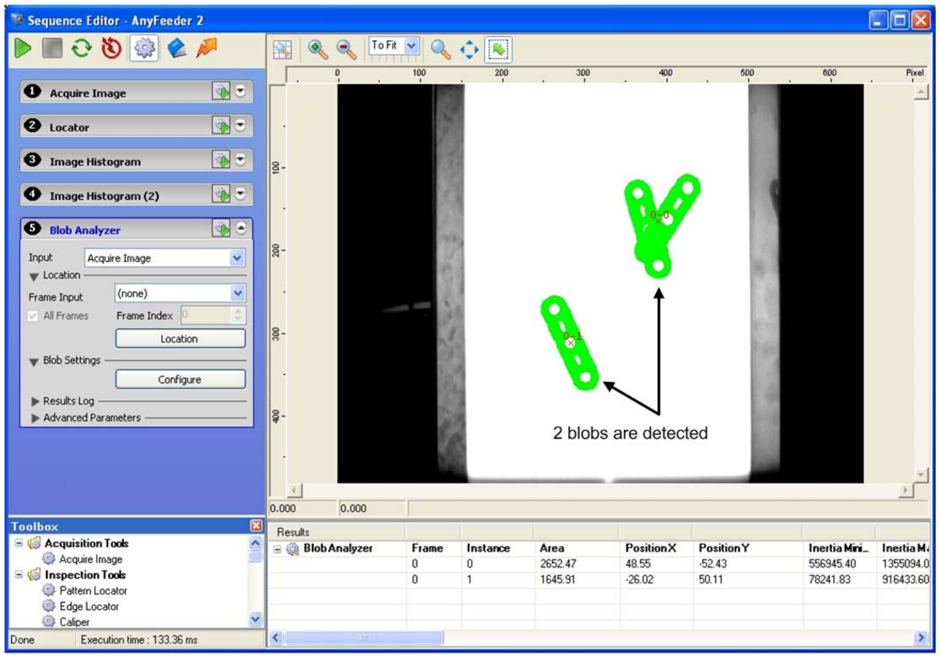


Input: Acquire Image
Frame Input: (none)
Frame Index: 0
Models: L, [✓ I], Non L, [✓ T]
Search
Scale: Nominal 1.00
Rotation: Range -180 180
Instances to Find: All
Min Model Recognition (%): 98

Locator	Frame	ID	Model Name	Model ID	Scale	Rotation	X
	0	0	I	1	1	-114.8687	-12.74717

Done Execution time : 133.36 ms

Fig. 5. Locator recognizes one object



Input: Acquire Image
Frame Input: (none)
Frame Index: 0

Blob Analyzer	Frame	Instance	Area	Position X	Position Y	Inertia Mini_	Inertia Ma
	0	0	2652.47	48.55	-52.43	556945.40	1355094.0
	0	1	1645.91	-26.02	50.11	78241.83	916433.60

Done Execution time : 133.36 ms

Fig. 6. Blob Analyzer detects two blobs

4. Parts feeding strategies

In order to successfully pick parts from the Feed Surface, I designed and implemented the fallowing strategy. This strategy has to resolve the following problems:

- robot picks only recognized graspable parts
- program exists when desired number of parts is picked
- if Feed Surface contains no parts, then parts are brought from the Bulk Container (Dispense)
- if Feed Surface contains unrecognized parts, then parts are Flipped
- if Feed Surface contains blocked parts and parts were picked, then it is inspected again to detect possibly freed parts
- if Flip is repeated too many times without detected parts becoming recognized, present parts are considered rejects and Purged
- if Dispense is repeated too many times without resulting in parts on the Feed Surface, the "No parts in Bulk Container" error is issued, program exits
- if parts are still on the Feed Surface after a Purge command, the "Rejects blocked on Feed Surface" error is issued, program exits
- if too many Purge commands are issued, then "Too many rejects" error is issued and program exits

After uniting the solutions to all of these problems, I designed the fallowing algorithm (also presented in Fig. 7):

Init: initialize variables
Step 1: Inspect Feed Surface, if it contains parts then go to **Step 3**, otherwise go to **Step 2**
Step 2: Dispense parts, if maximum number of successive Dispense commands is reached then go to **Errors**, else go to **Step 1**
Step 3: pick recognized graspable parts, if desired number of picked parts is reached then go to **End**, if Feed Surface contains blocked parts and parts were picked then go to **Step 1**, if Feed Surface contains only unrecognized / ungraspable parts then go to **Ste 4**
Step 4: Flip parts, if maximum number of successive Flip repetitions is reached then go to **Step 5,** else go to **Step 1**
Step 5: Purge rejects, if Feed Surface still contains rejects then go to **Errors**, else go to **Step 2**
Errors: report encountered error, go to **End**
End: end program.

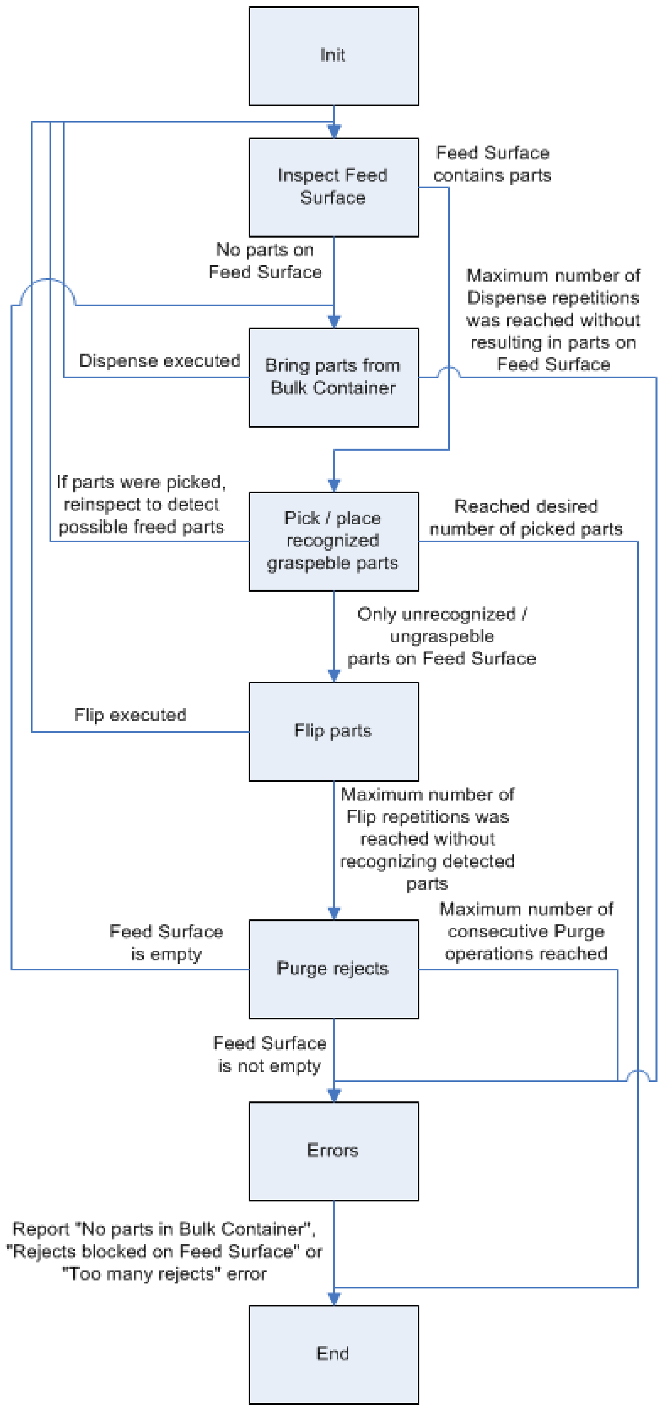

Init
Inspect Feed Surface
Feed Surface contains parts
No parts on Feed Surface
Maximum number of Dispense repetitions was reached without resulting in parts on Feed Surface
Dispense executed
Bring parts from Bulk Container
If parts were picked, reinspect to detect possible freed parts
Pick / place recognized graspeble parts
Reached desired number of picked parts
Only unrecognized / ungraspeble parts on Feed Surface
Flip executed
Flip parts
Maximum number of Flip repetitions was reached without recognizing detected parts
Maximum number of consecutive Purge operations reached
Feed Surface is empty
Purge rejects
Feed Surface is not empty
Errors
Report "No parts in Bulk Container", "Rejects blocked on Feed Surface" or "Too many rejects" error
End

Fig. 7. Part feeding strategy

5. Multitasking control

The workstation is integrated in a manufacturing cell. The hierarchy is based on the Master - Slave model. So, the cell's PLC is the Cell Master, the robot controller is the Workstation Master, while the two feeders and the vision software (which runs on the workstation's PC) are the Workstation Slaves (also see Fig. 8).

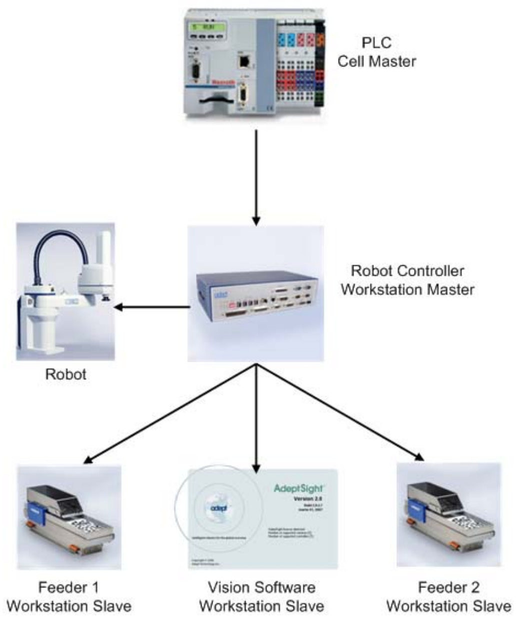


Fig.. 8. Workstation hierarchy

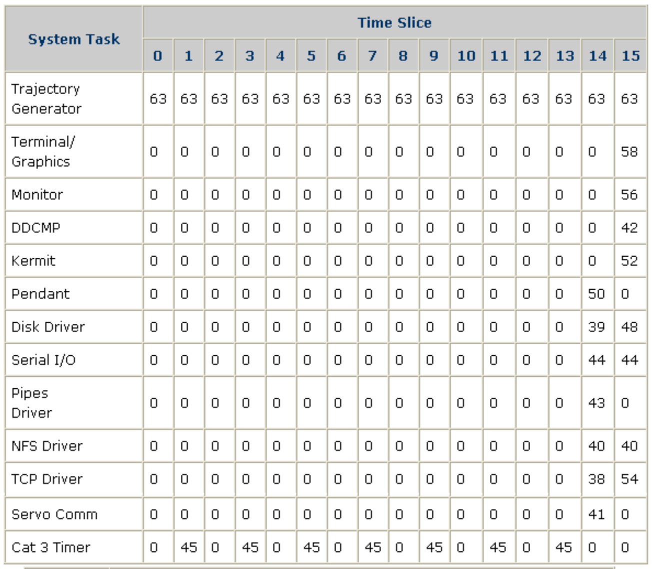
The robot controller features a multitasking industrial processor. Just like any other multitasking processor, this processor can execute a single task at any given time, but all tasks take alternatively control of the processor for very short periods of time, thus creating the impression that all tasks are running simultaneously. In order to decide which task deserves to run next on the processor, processing time is divided in a major time slices, each of these being 16ms long. Every major slice is divided in 16 equal minor slices. Each system or user task has a priority (ranging from -1 (do not run) to 63 (maximum)) assigned for each slice (system tasks priorities and user tasks default priorities are presented in Fig. 9). At the beginning of each minor slice, the processor makes a list of ready to run tasks and assigns control over the processor to the task with the highest priority, when this task finishes running, priority is assigned to the next task and so on until all tasks run or the slice ends.

For implementation, I will use 3 user tasks:

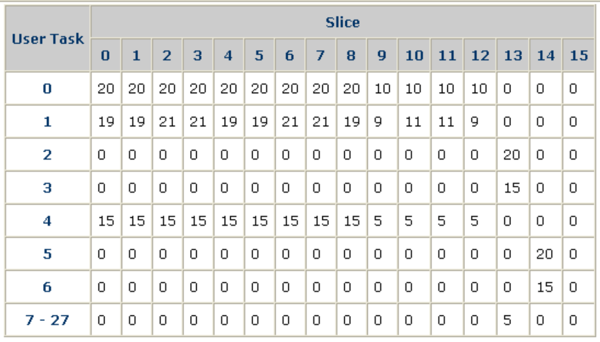
0. PLC communication task - this task will communicate with the PLC to obtain work orders and report work completion
1. Feeder 1 - this task will handle communication with Feeder 1 and graspable part choosing from this feeder, it will also move the robot to pick parts from Feeder 1
2. Feeder 2 - the same thing as with Feeder 1, but for Feeder 2

The task propriety settings will need to resolve the following problems:

- □ don't block the system tasks
- □ don't block communication with the PLC
- □ use the robot as much as possible
- □ give equal running time to Feeder 1 and Feeder2 tasks
- □ integrate a "critical region" mechanism so that Feeder 1 and Feeder 2 get alternative use of the robot, but one feeder is free to obtain parts as the other feeder is commanding the robot



System Task	Time Slice															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Trajectory Generator	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63	63
Terminal/ Graphics	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	58
Monitor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	56
DDCMP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	42
Kermit	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	52
Pendant	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0
Disk Driver	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	48
Serial I/O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	44	44
Pipes Driver	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43	0
NFS Driver	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	40
TCP Driver	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	54
Servo Comm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	41	0
Cat 3 Timer	0	45	0	45	0	45	0	45	0	45	0	45	0	45	0	0



User Task	Slice															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	20	20	20	20	20	20	20	20	20	10	10	10	10	0	0	0
1	19	19	21	21	19	19	21	21	19	9	11	11	9	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0
4	15	15	15	15	15	15	15	15	15	5	5	5	5	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0
7 - 27	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0

Fig. 9. Default task priorities

In order to implement solutions to these problems, the default user tasks priorities are not adequate. I will re-write these priorities, as in Fig. 10. In this way, PLC task always has highest user priority, Feeder 1 and 2 tasks have equal priority. Also, Feeder 1 and 2 tasks have lowest possible priority for the last two slices that are used by system tasks.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PLC	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
Feeder 1	29	29	29	29	29	29	29	29	29	29	29	29	29	29	0	0
Feeder 2	29	29	29	29	29	29	29	29	29	29	29	29	29	29	0	0

Fig. 10. Task priorities

In order for the robot to pick all available parts from one feeder and not "jump" between feeders when both have graspable parts, I will use a global variable that will tell the Feeder tasks which feeder is in control of the robot.

This variable is called "feeder_activ" and has the following values:

0. no Feeder has control of the robot
1. Feeder 1 has control of the robot
2. Feeder 2 has control of the robot

This functionality is also presented in Fig. 11.

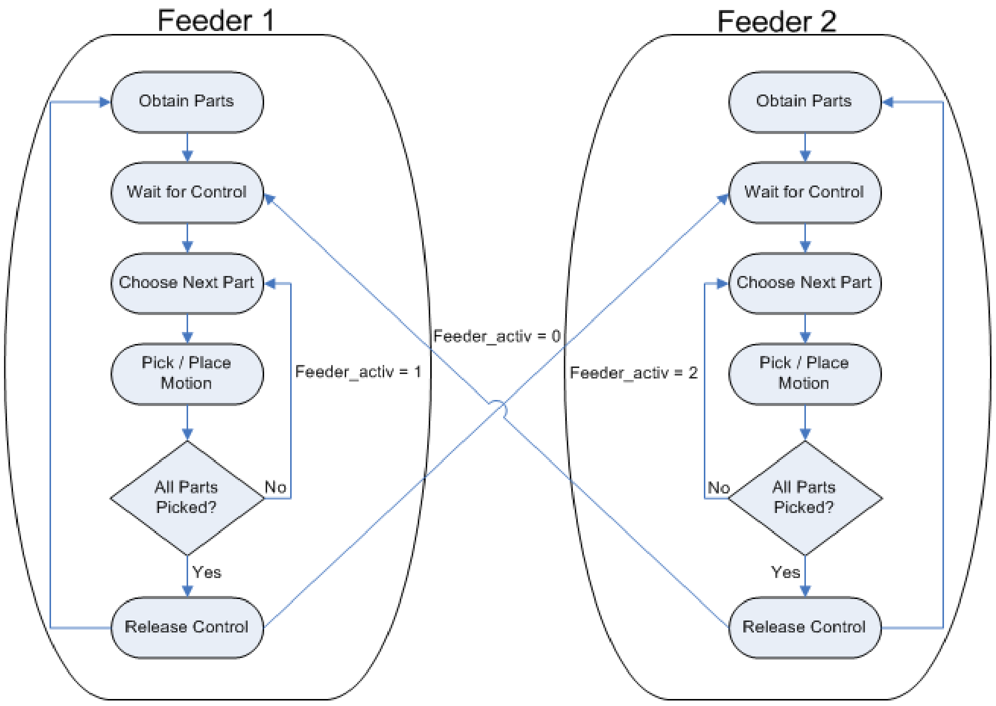


Fig.. 11. Tasks lock-out based on busy variable

In order to visualize how the workstation operates, we will assume that the time slices are small enough and the processor switches execution between the tasks fast enough so that tasks seem to run parallel and continuously. Also, we will assume that the workstation is commanded to produce 7 parts; 3 parts will be picked from Feeder 1, 2 will be picked from Feeder 2 and finally 2 parts will be picked from Feeder 1. In this case, Feeder tasks should run accordingly to Fig. 12.

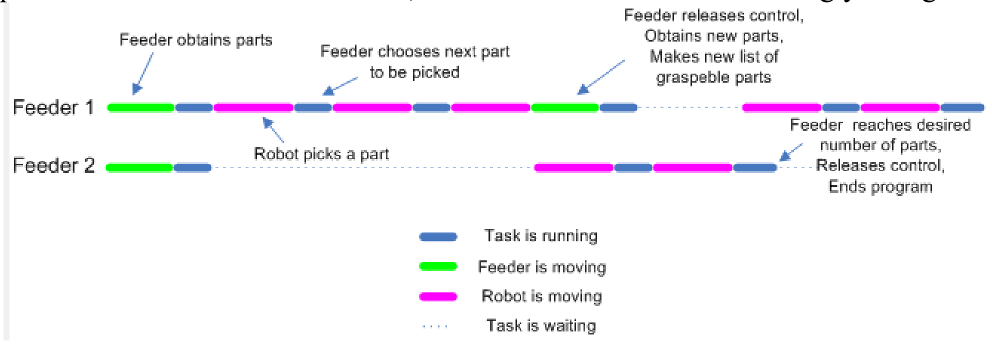

Feeder obtains parts
Feeder chooses next part to be picked
Feeder releases control, Obtains new parts, Makes new list of graspeble parts
Feeder 1
Feeder 2
Robot picks a part
Feeder reaches desired number of parts, Releases control, Ends program
Task is running
Feeder is moving
Robot is moving
Task is waiting

Fig.. 12. Task running diagram

6. Implementation and testing

In order to implement the workstation, I used a s800 SCARA robot, the two vibrating parts feeders and vision software from the same manufacturer; while a different manufacturer was used for the cell PLC and conveyor system The resulting workstation can be seen in Fig. 13.

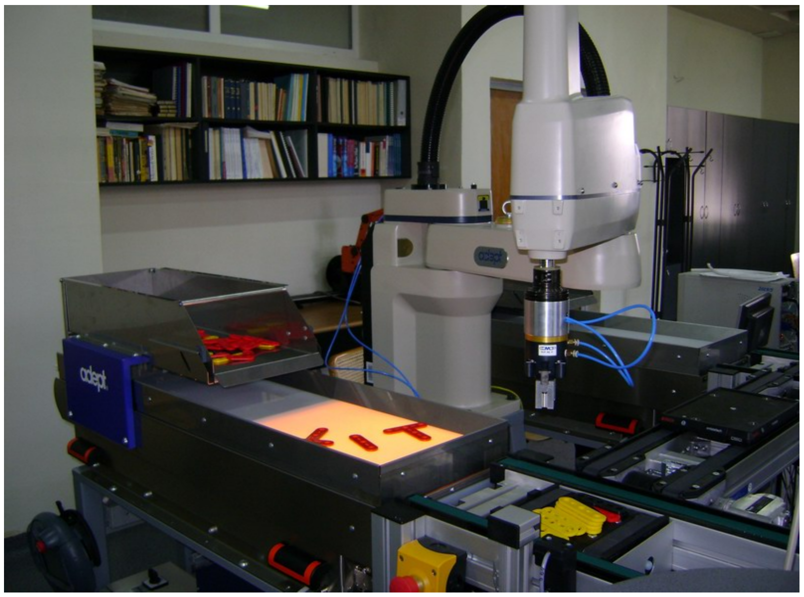


Fig.. 13. The Workstation

After the workstation was implemented, testing followed. The robot was commanded to pick 20 parts, first using only one feeder, then using both feeders. Both situations were repeated 10 times. Fig. 14 presents the total workstation working time and the robot working time percentage.

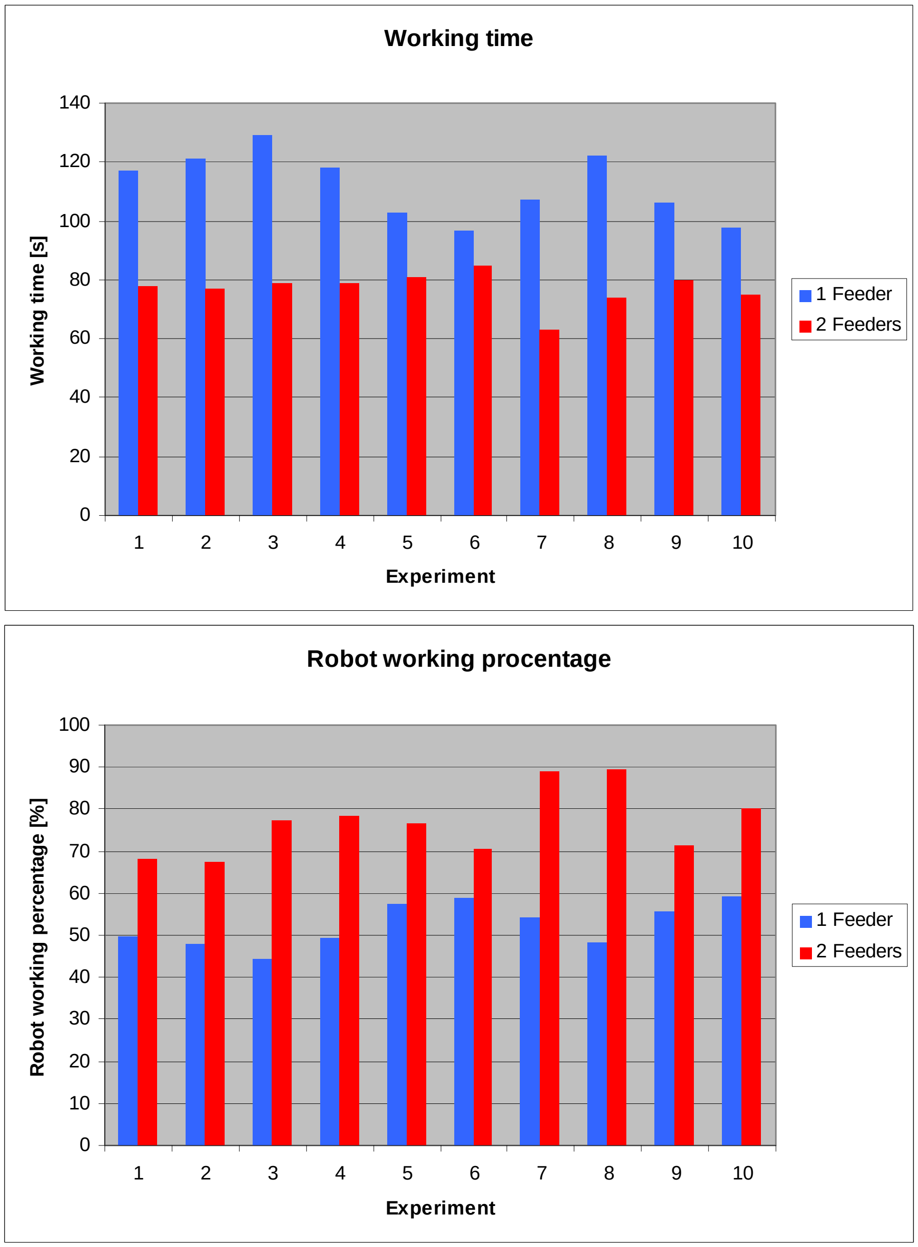


Fig. 14. Working time and Robot working time

7. Conclusions

As can be seen from Fig. 14, by adding a second feeder, we have succeeded in increasing the overall robot working time by about 20%. Also, as can be seen from Fig. 14, the total working time has dropped by about 30%. Another important aspect that can be seen in Fig. 14 is that the working time with 2 feeders spans a narrower range of values (approximately 20 versus 35 seconds), thus making the workstation's performances more consistent, easier to predict and easier to integrate in the production schedule.

On the price versus performance aspect, adding a second feeder to an existing workstation increases its cost with only 15%, while the performance increases with 30%. So, if only one feeder is present and performances are not satisfactory, the right thing to do is adding a second feeder, and not adding a second workstation.

REFERENCES

[1] *R. Barták,* Mixing planning and scheduling to model process environments, PACLP, Manchester, UK, 2000

[2] *G.G. Meyer, K. Främling, J. Holmström,* Intelligent Products: A survey, Computers in Industry, **Vol. 60**, Issue 3, April 2009, Pages 137-148

[3] *Th. Borangiu, F.D. Anton, S. Tunaru, A. Dogar, N. Ivanescu*, Robot-Vision Based Part Conditioning for Flexible Feeding Devices , Proc. of the 1st International Conference on Optimization of the Robots and Manipulators – OPTIROB 2006, May 26-28, Predeal, Romania

[4] *Adept Robotics*, Adept AnyFeeder User’s Manual

[5] *Th. Borangiu, A. Dogar, S. Tunaru, F.D. Anton, N. Ivanescu*, Vision Guided Robotic Grasp Learning Procedure, Proc. of the 1st International Conference on Optimization of the Robots and Manipulators – OPTIROB 2006, May 26-28, Predeal, Romania

[6] *M. Dragoicea*, Programarea aplicatiilor in timp-real. Teorie si practic (Applications Programming in Real-time.Theory and Practice – in Romanian), Editura Universitara, 2009, ISBN 978-973-749-579-2, cap. 2 si 3, pag. 87 - 139

[7] *Th. Borangiu, S. Raileanu, A. Rosu, M. Parlea*, Holonic Robot Control for Job Shop Assembly with Dynamic Simulation, Programmable Logic Controller, In-Tech Publications, Vienna, Austria, ISBN 978-953-7619-63-3, 2010

[8] *S. Raileanu,* Production scheduling in a holonic manufacutring system using the open-control concept, U. P. B. Sci. Bull., Series C, **Vol. 72**, Iss. 3, 2010, pag. 39-53, ISSN 1454-234x

[9] *Th. Borangiu et al*, Industrial Robotics: Theory, Modelling and Control, Advanced Robotics Systems, Vienna, Pro Literatur Verlag Robert Mayer-Scholz Germany, Ed. Sam Cubero, ISBN 3-86611-285-8, 2006