Reconfigurable Distributed Controller for Welding and Assembly Robotic Systems: Issues and Experiments

Alan Maldonado-Ramirez, Ismael Lopez-Juarez and Reyes Rios-Cabrera

Abstract Industrial production systems for smart factories or the so-called Industry 4.0 will demand high interoperability and connectivity between production modules, so that modules could be monitored in real-time. Production modules should make decisions on their own without human intervention; and they must be modular and adaptive to changing circumstances and customers' requirements. The autonomous operation of production modules in smart factories imposes asynchronous delays due to several reasons, such as object recognition time, grasping time or welding delays that change due to a newly reoriented or positioned component. Consequently, production modules need to be speeded up to compensate for the delays in the previous production stages. In this paper, we present a novel Reconfigurable Distributed Controller (RDC) for Intelligent Robotic Welding and Assembly Systems that autonomously compensate the production delays. The proposed RDC compensates for three types of major production delays that affect the total production time. (I) The first delay can occur at individual level. In this case, the module can fully compensate, since no other modules are affected and the total production time for this product can be met. (II) The second type of delay occurs at inter-module level, where delays are so long that more than one production module will need to be reconfigured. (III) Finally, the third type of delay occurs in the worst-case scenario when the total production time cannot be met by modifying individual module's production time. A total cell reconfiguration is needed, which implies to speed up the next production cycle to deliver the following product before its deadline. By doing so, the mean production time is maintained. In this paper, issues and experiments that show the feasibility of the RDC are presented. Results of using a distributed reconfigurable manufacturing cell composed of three industrial robots, conveyor belts, and a positioning table demonstrated the effectiveness of our approach to compensate the major delays in real working environments.

A. Maldonado-Ramirez · I. Lopez-Juarez (✉) · R. Rios-Cabrera
Robotics and Advanced Manufacturing, CINVESTAV—IPN,
Ramos Arizpe 25900, Mexico
e-mail: ismael.lopez@yahoo.co.uk

S. Chen et al. (eds.), *Transactions on Intelligent Welding Manufacturing*,
Transactions on Intelligent Welding Manufacturing,
https://doi.org/10.1007/978-981-10-8740-0_2

Keywords Reconfigurable distributed control · Smart factories
Intelligent robot welding · Intelligent robot assembly

1 Introduction

The advances of science and technology are continuously supporting the improvement of the daily products and services, and manufacturing technology is not the exception. All these developments have led to the so called fourth industrial revolution or industry 4.0 with the introduction of the smart factories and the Internet of Things (IoT) paradigm.

As the complexity of the manufacturing system grows, due to the inclusion of robots, CNC machines, conveyor belts, sensors and other equipment, the manufacturing cell requires more flexible control strategies that are not reachable with a Programable Logic Controller (PLC) as the central control element. On the other hand, a distributed control system, where all the hardware and software components connected, it improves the communication and coordination of all the elements through the passage of messages to achieve a common goal.

The introduction of the DCS allows the implementation of flexible and versatile control strategies. By using intelligent production schedulers, we can compensate perturbations caused by asynchronous faults, without limiting the working capacity of the manufacturing cell. It can also implement on-line fault correction technics and control schemes for intelligent specialised modules such as assembly robots with force-torque sensing and a welding robot with learning skills.

In order to develop a manufacturing system based on the Industry 4.0 and IoT strategies, it is necessary to use equipment compatible with high level communications schemes. For this project, we use KUKA Industrial Robots able to communicate via ethernet with the TCP/IP protocol, high speed wireless networks routers, GigE industrial cameras and high specs micro-computers.

The organization of this paper is as follows. Section 2 present the most relevant related work and original the contribution of this paper. Sections 3, 4 and 5 describe the communication framework, the computer vision specialised module and the distributed architecture of our flexible manufacturing system. Section 6 gives a detailed explanation of the experimental work carried out in a real manufacturing cell integrated by industrial equipment, as well as the performance results of our methods. Finally, Sect. 7 provides the conclusion and the envisage future work.

2 Related Work

One of the core parts of this work was the definition of a communication framework. Since there are many ways to establish an open communication with a robot controller, one of the most popular interfaces is the Robot Operating System

(ROS) [1]. Unfortunately, KUKA industrial robots do not fully support ROS yet. A solution to this problem was found in Ref. [2]. There authors propose a server called KUKAVARPROXY and JOpenShowVar, and it has several advantages such as the low cost, flexibility, reliability and integrability.

In terms of Industry 4.0 and the IoT paradigm, the most important characteristics of a smart factory are the mass production customization, flexibility, self-reconfiguration of the working elements. It is also important the global knowledge of the manufacturing system status, in order to optimize the decision making in a real-time monitored supply chain [3]. Industry 4.0 is also related to the concept of cloud manufacturing. This is a model that is service oriented, customer centred and demand driven. In Refs. [4, 5] the authors explored the potential future of this manufacturing model and how it can be implemented to automation and industrial control systems with a strategic point of view based on the cloud

The introduction of the cloud manufacturing model, implies the design of optimal scheduling algorithms. In Ref. [6] a multi-objective optimization scheduling model was introduced to improve the production efficiency of a reconfigurable assembly line, particularly balancing the production load and minimizing the delayed workload.

In Ref. [7] Georgios et al. described the application Information Technologies such as design and manufacturing. They also examined the use of cloud computing in the mechanical drawing and design process of an enterprise. It proposes a specific architecture with different servers, for the implementation of a collaborative cloud-based Design system. It also compares the operating cost of an industry's design department before and after the use of the proposed system. However, this paper focuses mainly on software and interactions but not on real manufacturing cells.

In Ref. [8] Pei et al. dealt with the problem of jobs characterised by non-identical sizes, different release times and unequal processing times. The objective is to minimise the makespan by making batching and sequencing decisions. The authors formalized the problem as a mixed integer programming model and they showed the problem to be strongly NP-hard. Some structural properties are presented for both the general case and a special case. The main focus of the paper is algorithmic. In Ref. [9] it is investigated the coordinated scheduling problem of production and transportation in a two-stage supply chain, where the actual job processing time is a linear function of its starting time. They considered that in the production stage the jobs are first processed in serial batches on a bounded serial batching machine at the manufacturer's site. The main contribution is the development of an optimal algorithm to solve the problem of minimizing the makespan. However, the real implementation is not part of the scope of the paper.

As the mass production customization arises, so does the necessity to implement reconfigurable manufacturing systems. In contrast to the traditional manufacturing systems RMS can change their components, structure and interactions over time in order to rapidly change its production mode to manufacture different items from the same family. In Ref. [10] the authors presented a formal modelling and analysis method to describe the behaviour and verify the reconfiguration of an RMS.

The concept of smart factories does not only include the characteristics listed above. It also includes the incorporation of smart processes, such as a specialised intelligent assembly module as reported in Ref. [11], where the authors employed a force-torque sensor to provide important contact information to a neural controller dedicated specifically to that assembly task. Industry 4.0 can also include intelligent welding modules with learning skills such as the proposal of Ref. [12].

In Ref. [13] an architecture for a flexible manufacturing system was presented. This integrated a distributed control system, endowing some of the working elements with reconfiguration capabilities. This work also included a production controller with scheduling algorithms such as Early Deadline First and Deferrable Server. However, this system was limited in its reconfiguration capabilities.

2.1 Original Contribution

The aim of this paper is to introduce a novel architecture of a Flexible Manufacturing System. Our proposal includes a Reconfigurable Distributed Controller implementing a fault correction algorithm, capable of compensating the production delays. We also proposed a communication framework superior to Ref. [2], based on a C++ JOpenShowVar variation, allowing easy integration of new modules to the manufacturing system. Our proposal meets the Industry 4.0 and IoT paradigms, endowing the system with full-reconfiguration capability.

3 The Client/Server Implementation

There are many ways and protocols to establish a communication between different devices. If we use computers, the most common method is the TCP/IP protocol. This type of computer networks and communication protocol allow different applications to communicate with each other, and the parts can be used to achieve a common goal. If the application consists of two or more parts running on different devices and communicate with each other to solve a common task over a computer network, it is called a distributed application.

The implementation of a distributed application grants a series of benefits compared with an application run on a single computer, as showed in The Distributed Architecture Sect. 5. Particularly for a manufacturing system, it allows the development of Distributed Control Systems (DCS). In order to program our client/server applications, the open source libraries of Boost.Asio for C++ were used.

In general, there are 2 important communication protocols in network programming, the TCP protocol and the UDP protocol. Each one of them have different characteristics suitable for a particular type of application. In the case of our distributed application we opted to use the TCP protocol for the following reasons:

- It guarantees delivery of the message, including error handling mechanisms in the case of a failure in the communication.
- It establishes a point-to-point communication model.
- Before the communication can take place over a TCP protocol, a logical connection must be established by exchanging service messages.

3.1 The Client Application

To develop the client application, there are two options:

- A synchronous TCP client.
- An asynchronous TCP client.

The synchronous TCP client waits until it gets a response from the server, while the asynchronous one can execute another action while it waits for the server response. In our particular application, the time that it takes the client to send a request and to process the server response, is very fast. This is because the amount of data sent was designed to be small. For this reason there is no need to execute parallel actions while the client is waiting for a response. Therefore we decided to use the synchronous TCP client.

3.2 The Server Application

In terms of server applications, they can be classified according to the way they serve clients. Focusing only on the synchronous servers they can be:

- An iterative TCP server.
- A parallel TCP server.

An iterative server can only serve one client request at the time, while the parallel one can process more than one client request, making use of multi-thread programming techniques.

The distributed application focused in our work consist on many servers and clients. That is why it was necessary to implement servers able to handle many clients at the same time. For this reason, the parallel TCP server was chosen.

The Boost.Asio libraries used are multi-platform and we were able to program all client/server applications on Raspberry Pi microcomputers. Each module was configured to be in charge of controlling a specialised task, in the distributed manufacturing system, sharing its information through the communication network.

4 The Computer Vision Specialised Module

In smart manufacturing, of the most important aspects to take into account are the sensors. The introduction of different kind of sensors in a manufacturing system provides the controllers with valuable external information to make important decisions.

The sight sense is one of the most important senses of humans, since it provides information about the surroundings such as:

- Extract position, orientation distance and size of objects.
- Know if an element is static or moving.
- Compare characteristics between a set of objects.
- Among others.

To ensure that these skills in a manufacturing system can be achieved, we must rely in the usage of cameras. Including computer vision in a specialised module allows more flexibility for robotic tasks such as handling, welding, assembling, among others. It can also help to minimise errors caused by a badly located element and it also allows quick modifications to the manufacturing process without programming changes.

In our specialised module, we integrate the advantage of the current industrial technology in computer vision, combined with or own developed programs based on open source solutions. Technologies such as the GigE Vision camera standard and the Pylon SDK developed by Basler AG to create camera applications together with computer vision libraries from OpenCV.

It orders to detect and grasp objects, it was necessary to design and create our own manufacturing tools for the industrial robots. We made use of Rapid Prototyping Technologies to 3D print our tool models in order to test them in the manufacturing system. Figure 1 shows our designed gripper, and the location of the camera.

Fig. 1 Developed gripper for handling and assembly tasks and the camera is located near the grasping area

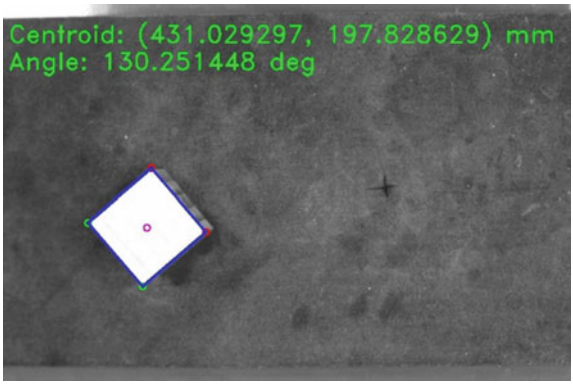

Fig. 2 Centroid and orientation extraction of a sample object, and an industrial robot uses this information to perform different tasks

This gripper (Fig. 1) was developed for a KUKA industrial robot to execute handling and assembly tasks aided by computer vision. An important aspect of using the GigE Vision industrial standard is that it allows us to get access to camera information remotely within the local network to process it, in the computer vision specialised module, or any other available module.

With the computer vision specialised module, we can perform image processing online in a remote computer, extract the valuable information from the image and then share it with the robot for it to execute different tasks (grasping, moving, welding, assembly, etc.). Figure 2 shows an example of extracting coordinates of the piece centroid and orientation angle, for the robot to perform a grasping action.

Running this module on a remote CPU in a local network, it grants the advantage of easy integration to a distributed manufacturing system. This feature is mainly achieved by the utilisation of GigE vision industrial standard and the Basler AG Pylon SDK to create our own client/server application to access the camera information.

5 The Distributed Architecture

Flexible Manufacturing Systems can be divided into 2 types of distributed architectures:

- The Distributed Control System (DCS).
- The Manufacturing Tasks Distribution.

These two types of distribution are highly valuable in the future of manufacturing systems. The use of a DCS increases the reliability and security of operations, not depending in just one central controller but in a network of controllers that can performed the supervising function in case of failure.

In our platform, each module or working elements can fulfil the same series of tasks or functions (redundancy in functions). Therefore, the manufacturing process

can be distributed in a network of working elements. This distribution in the manufacturing process implies advantages such as:

- Reduction in the production time.
- Relieve in specific production task in order to avoid bottlenecks.
- Scheduled maintenance to specific elements without stopping the production.

In the next subsections, a detailed explanation of these two distributed architectures is given.

5.1 The Distributed Control System Hardware

The development of embedded computers and the increasing power of this microcomputers able of internet connection, is pushing forward the implementation of Internet of Things (IoT) technologies in many areas, particularly in the Industry 4.0. The utilisation of microcomputers in the control of a manufacturing system grants the capability to implement more complex production strategies, such as dynamic scheduling with real-time decision-making algorithms that contemplate the system status the whole time.

In order to get a distributed control system that does not depend on PLCs we can develop applications in C++, Java or any other programming language able to run on embedded computers.

Since our flexible manufacturing cell is mainly integrated by KUKA industrial robots, we opted to develop the communication framework based on the open source KUKAVARPROXY server application. With this server application running on each of the robot controllers, we can read/write system variables as well as user defined variables. With this, we can translate the whole robot control to a smart distributed module.

As shown in Fig. 3 a client application is programmed in a Raspberry Pi microcomputer in order to supervise the robotic system status. In this microcomputer, we can also include a production controller and a decision-making algorithm based on the full system status.

For the DCS, a network of microcomputers is established with a client/server communication framework. The principal function of this computers is to control each of the modules of the flexible manufacturing system (welding, assembly, milling, handling, computer vision and conveyor belts) as shown in the Fig. 4.

Each microcomputer is connected to the local network running a client/server application. In this way, all the computers can share the status of their corresponding specialised modules. This is a very important feature of the DCS, because here lies the reliability, in the network of controllers. Knowing the status of each microcomputer, in the case of a server failure, its supervising function can be relieved to another microcomputer in the network, as shown in Fig. 5.

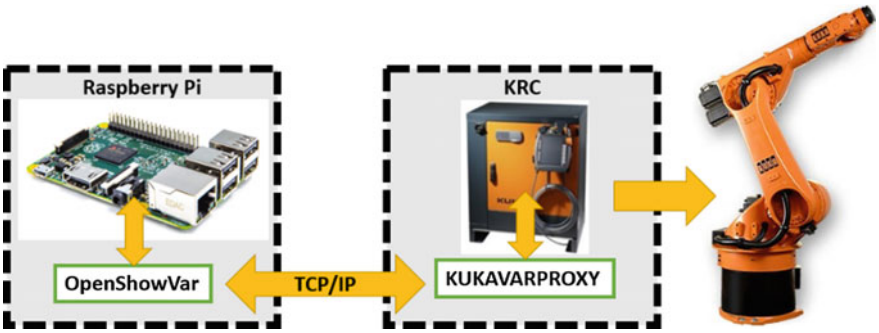

Fig. 3 KUKAVARPROXY communication framework and the robot control can be performed in the distributed module

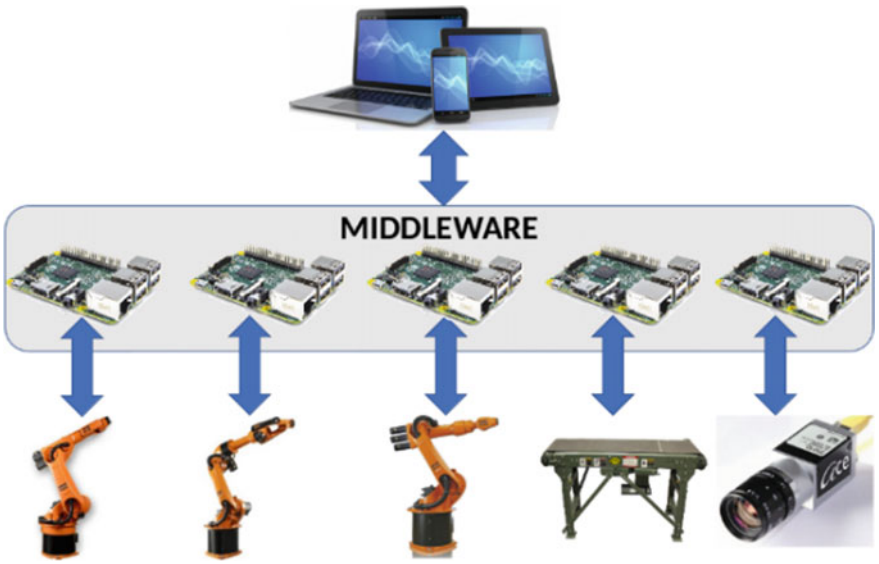

Fig. 4 Distributed control system structure

The introduction of a DCS with the reliability advantages that it grants, does not exclude the need of a human supervising function. For this reason, it is important the design of a Human–Machine Interface (HMI) to provide the user with the most valuable information about the system status.

The HMI shown in Fig. 6 was designed to help a human with the supervision of the whole manufacturing cell. This HMI shows the individual status of each of the specialised modules and allows the user to configure the production parameters.

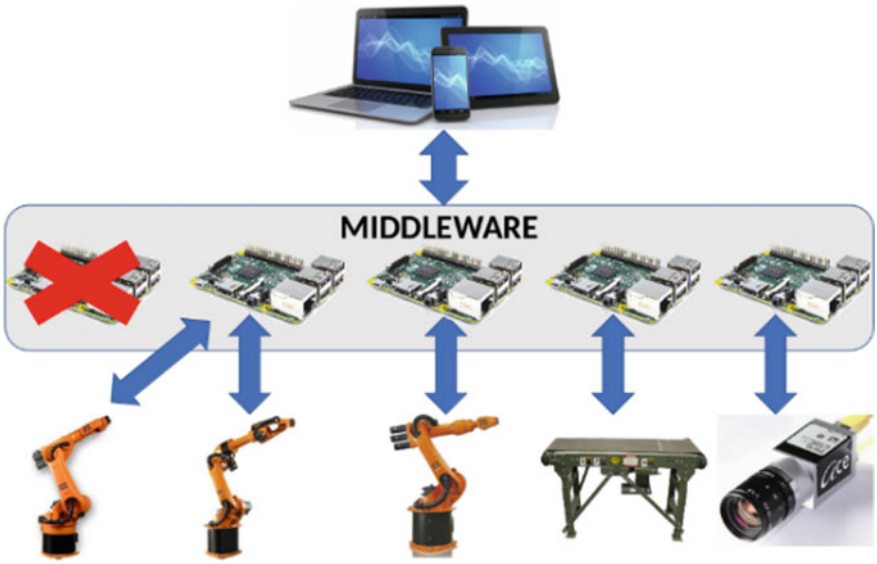

Fig. 5 Failure in a microcomputer of the DCS

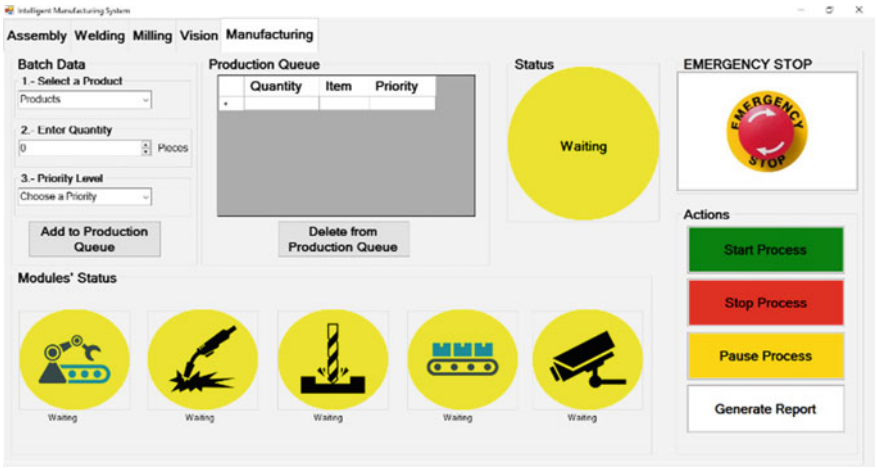

Fig. 6 Designed HMI to control the distributed manufacturing system

5.2 The Manufacturing Tasks Distribution

As mentioned before, if we want to make a truly reliable distributed manufacturing cell, it is required to have modules that can fulfil the function of not only one specialised module. In our case, we need to enable robots to execute more than one task. We defined four: handling, welding, assembly and milling.

Table 1 Set of functions per robot

Robot	Functions			
	Assembly	Milling	Handling	Welding
KR60	Y	Y	Y	–
KR16HW	Y	–	Y	Y
KR16	Y	Y	Y	–

Our current experimental platform is integrated by 3 KUKA industrial robots: KUKA KR16, KUKA KR60 and KUKA KR16HW. Each of these robots can execute a different set of functions as shown in Table 1.

This robot capability to execute a different set of tasks, allows us to test the distribution of procedures in the manufacturing process that implies assembly, milling, handling or welding.

This distribution of tasks provides 2 key features.

Intelligent Production Controllers.

The first benefit is the capability to implement intelligent production controllers to optimize the manufacturing time. This means, the manufacturing process will not be a strict steps sequence that must be executed by a specific robot, but a dynamic process where any of the available robots can perform different steps of the manufacturing process.

For example, let's consider a simple manufacturing process with the next sequence:

The raw material enters the production line.

- Milling process.
- Assembly process.
- Welding process.
- Finished product.

Now, in the traditional production line this process would be performed by 3 robots where each one of them would execute one step from 2 to 4, as shown in Fig. 7. In this approach, the system is sensitive to the delays that may occur in the specialised modules, giving rise to bottlenecks in the production line.

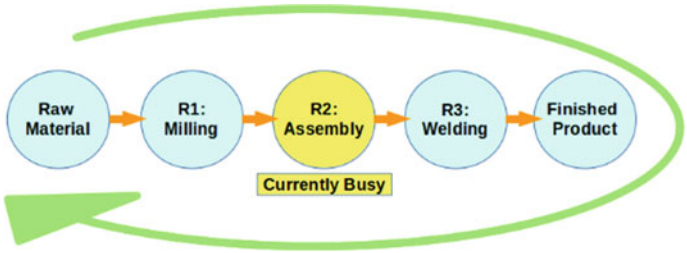

Fig. 7 Traditional continuous production line

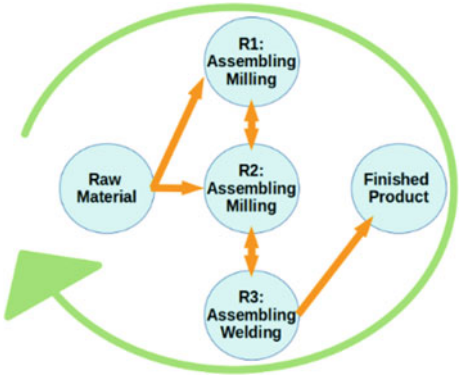

Fig. 8 Distributed tasks in the production line

The focus or our proposal aims to give a solution to this kind of problems with a distribution of tasks. In order to avoid the bottlenecks in the manufacturing line, the global production controller takes into account the status of the whole system to assign the task to any of the available robots, as shown in Fig. 8.

Robustness to Major Module Failure.

The second important advantage is the robustness to major module failure. Similar to the distributed control system where a computer in the network fails, another one relieves its supervising functions, if a robot fails in the distributed manufacturing system, the production could still continue, because another robot can relieve the pending tasks.

This feature may not only apply in case of failure but also for scheduled maintenance when a particular element of the production chain must be stopped. Taking the same example of Fig. 8, if the robot R1 with the assembly and milling functions is stopped, the robot R2 can accomplish the same tasks and continue with the production.

In our proposal all these characteristics of flexible manufacturing system are achievable based on the proposed communication framework. Our DCS running on microcomputers capable of monitoring the status of the entire system and a dynamic production controller suited to modify the distribution of task according to the acquired knowledge of the system status.

5.3 The Fault Compensation Algorithm

Unlike the classical manufacturing systems where each step in the process is strictly coordinated to start and finish in a specific time in a rigid structure, the application of intelligent modules implies the existence of non-deterministic processing times in the specialized functions.

As a third part of the distributed architecture, a fault compensation algorithm is included in this work. The main function of this algorithm is to change dynamically the working speed of each individual element of the manufacturing system in order to compensate the existence of delays due to failures or non-deterministic processing times.

This algorithm is based on the previous knowledge of the mean execution time for each of the task that are involved in the manufacturing process and the minimum execution time that can be achieved by increasing the working speed of every task.

With the aid of the DCS described before, the execution time measurement of all the tasks can be easily done. And with the database of the mean execution times it is possible to identify in which task a delay has occurred. It is also important to consider a limit in the increment of the working speed to avoid undesired effects in the manufacturing process. In this context, the algorithm function is to speed up, in a safe way, the incoming task after the delay, until it is fully compensated. After that the system continues at is nominal speed, as described below:

```
if DT > 0 then
  if DT > MC then
Vel = Max_value
DT = DT – MC
  else
Vel = vsf(DT)
  end if
else
   Vel = DEFAULT
end if
```

where *DT* is the accumulative delay time in seconds; *MC* is the maximum time compensation in a task; *Vel* is the working speed of a module; *vsf(DT)* is a velocity selection function according to the delay time.

This function works with the database that describes the velocity/time relationship of the evaluated task.

6 Experiments

The experiments to validate the proposed Distributed Control System and the task distribution controller with the Fault Compensation Algorithm where carried out in the Intelligent Manufacturing Laboratory from CINVESTAV Campus Saltillo, showed in Fig. 9. The experimental platform used for experimentation is integrated by:

Fig. 9 Intelligent manufacturing laboratory CINVESTAV Saltillo

- 1 KUKA KR16 Robot with KRC2 controller.
- 1 KUKA KR16HW Robot with KRC4 controller.
- 1 KUKA KR60 Robot with KRC4 controller.
- 1 KUKA DKP400 2 DoF table.
- 1 KUKA KL1000-2 linear unit
- 1 Fronius TPS4000 welding module.
- 4 Raspberry Pi 3 microcomputers.
- 3 Basler Aca1300-gc GigE cameras.
- 3 Hytrol conveyor belts.

All this equipment is connected to the local area network in order to implement the DCS. The mentioned robots are equipped with the corresponding tools to execute the specialised functions described in Table 1.

In order to test the intelligent manufacturing cell at its full capacity, the experimental product shown in Fig. 10 was designed. The manufacturing process of this product implies handling, milling, assembly and welding tasks that will be carried out by the 3 different robots. This is, the KR60 robot picks the working piece from the first conveyor belt then takes it to a working area where it mills the

Fig. 10 Designed experimental product

holes. After that, this robot takes the milled piece to the second conveyor belt, where the KR16HW will pick it up and place it on the DKP400 positioner. The KR16 robot will go to warehouse, pick the cylinders and assembly them in the milled piece on the DKP400. The KR16HW will continue with the welding process, fixing the cylinders to the milled piece. At the end, the KR16 will pick the finished product from the DKP400 and place it on the third conveyor belt.

In a general way, the manufacturing process of this product is integrated by 22 task, that are listed in Table 2 with its corresponding mean execution time and the maximum time compensation that can be achieved with the working speed adjustment.

From Table 2 we can see, that the mean production time of one product is 218.268 and the maximum delay that can be compensated is 104.208 s in a full production cycle, corresponding to the 47.74% of the total manufacturing time. As it can be inferred, as the production process advance, the delay time that can be compensated is reduced, leading to the next 3 study cases:

- **Module Reconfiguration**: This happens when a delay has been detected in one task, and the same module where the delay occurred can fully compensate it.

Table 2 Manufactuing process description

Task	Robot	μT (s)	MaxCompensation
1. Go to CB1	KR60	9.72	4.716
2. Pick from CB1	KR60	18.912	9.168
3. Go to working area	KR60	7.284	3.492
4. Milling	KR60	16.92	8.16
5. Pick from working area	KR60	16.92	8.16
6. Go to CB2	KR60	12.624	6.168
7. Place in CB2	KR60	18.816	9.168
8. Return to home position	KR60	12.672	6.192
9. Go to CB2	KR16HW	7.332	3.516
10. Pick from CB2	KR16HW	8.304	3.888
11. Go to DKP400	KR16HW	7.356	3.516
12. Place in DKP400	KR16HW	18.552	8.82
13. Go to warehouse	KR16	4.776	2.232
14. Pick cylinder from warehouse	KR16	8.304	3.888
15. Go to DKP400	KR16	4.812	2.268
16. Assembly	KR16	7.512	3.48
17. Welding	KR16HW	17.388	8.004
18. Pick from DKP400	KR16	7.548	3.516
19. Return to home position	KR16HW	6.672	3.192
20. Go to CB3	KR16	8.268	3.996
21. Place in CB3	KR16	8.712	4.08
22. Return to home position	KR16	8.208	3.972

- **Inter-Module Reconfiguration**: When the delay is longer than the compensation that one single module can execute. This reconfiguration speeds up the next modules to complete the compensation.
- **Inter-Production Reconfiguration**: As the worst-case scenario, this happens when the delay is larger than the entire system time compensation or the delay is detected in one of the last steps in the manufacturing process. In this case the system will not be able to compensate the delay to deliver the finished product in time, so it will speed up the next production cycle to deliver the next product before time, to keep the mean production time.

6.1 Module Reconfiguration, Minor Delay in KR60 Computer Vision

This is the first case study, where a minor delay is detected in one of the first task of the KR60 robot. For this we are simulating a minor failure in the computer vision specialised module that tells the robot where to pick up the working piece from the first conveyor belt in the step number 2 of the manufacturing process. The applied delay was of 21.408 s in the mentioned task, assuming that it was the time for the computer vision specialised module to find the working piece and extract the position and orientation for the robot to pick it up.

Figure 11 shows how after the 21.408 s delay detected in the task number 2, the tasks from 3 to 5 speed up to compensate most part of the delay, and then the tasks

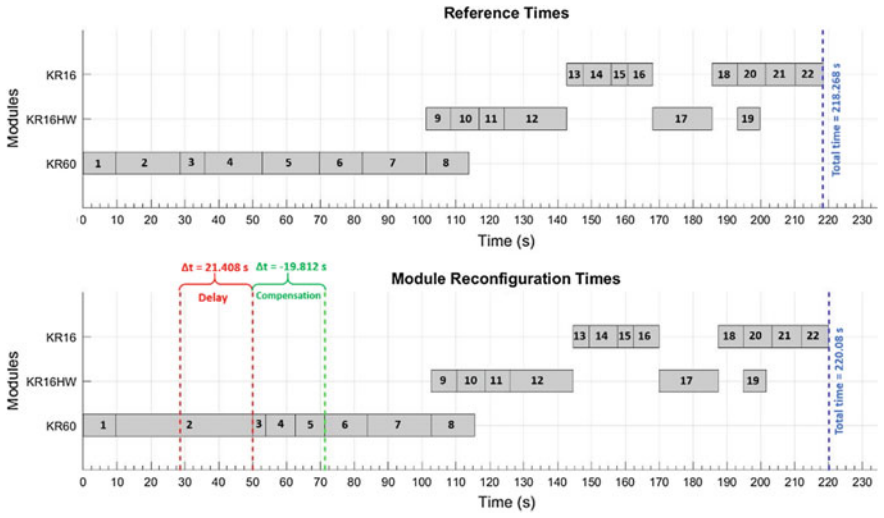

Fig. 11 Module reconfiguration, the delay shown in red, speeded up tasks in green

Table 3 Module reconfiguration times

Task	μT (s)	Actual time (s)	Δt (s)	Compensation (s)
1	9.72	9.72	0	0
2	18.912	40.32	21.408	0
3	7.284	3.792	17.916	−3.492
4	16.92	8.76	9.756	−8.16
5	16.92	8.76	1.596	−8.16
6	12.624	12.624	1.596	0
7	18.816	18.816	1.596	0
8	12.672	12.672	1.596	0

from 6 to 22 are fulfilled at the mean task time. Table 3 shows detailed information about the times, delay, and compensation time from task 1 to 8 that corresponding to the KR60 task to show the module reconfiguration principle. At the end, the system could not compensate 1.812 s, corresponding to the 8.46% of the total delay or to the 0.83% of the total production time. Nevertheless, the fault compensation algorithm minimized a delay of 8.46% from the total time to a 0.83%.

6.2 Inter-module Reconfiguration, Deterioration in the KR60 Milling Tool

This is the second study case, where a major delay is detected in the milling task of the KR60. For this, we are simulating a delay caused by the deterioration of the milling tool of the KR60 robot. This deterioration requires the robot to take a longer time to fulfil the task number 4 of the manufacturing process. For this experiment to test the inter-module reconfiguration the measured delay was of 51.144 s.

In Fig. 12 it can be seen how after the detected delay of 51.44 s in the task number 4 corresponding the milling operation of the KR60, the tasks from 5 to 16 were speeded up in order to compensate the whole delay. The inter-module configuration is observed, because tasks from 5 to 8 correspond to the KR60 robot, and from 9 to 16 correspond to the KR16HW and KR16 robot. Another thing that can be noticed in this experiment is that the fault compensation algorithm does not increase the speed of the tasks that do not contribute to the delay reduction, such is the case of the task number 8, KR60 returning to its home position, because the finishing of that task is not related with the start of task number 9. The corresponding compensation times for each robot are shown in Table 4. In this case we can see from the table that the algorithm made an over compensation of 2.916 s to finish the product in 215.352 s.

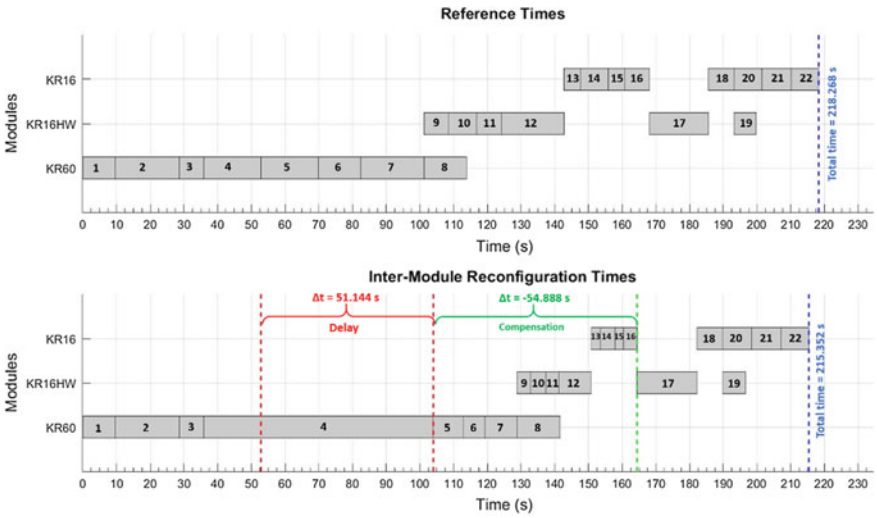

Fig. 12 Inter-module reconfiguration, delay shown in red, speeded up tasks in green

Table 4 Inter-module reconfiguration times

Task	μT (s)	Actual time (s)	Δt (s)	Compensation (s)
KR60	101.196	128.844	51.144	−23.496
KR16HW	58.932	39.756	21.408	−19.176
KR16	58.14	46.752	17.916	−11.388
Total	218.268	215.352	51.144	−54.06

6.3 *Inter-production Reconfiguration, Welding Parameters Change*

As the last study case, where a major delay is detected in the welding operation of the KR16HW. This case contemplates the inter-module reconfiguration because it is a major delay, but also it happens in one of the last tasks in the manufacturing process. For this, we are inducing a delay "caused" by the change in the welding parameters, such as the path velocity of the robot in the welding seam is changed. This change requires the KR16HW robot to take a longer time to fulfil the task number 17. In order to test the inter-production reconfiguration a delay of 59.796 s will be applied to task number 17.

As this is a major delay that was detected in one of the last steps in the manufacturing process, the fault compensation algorithm will not be able to fix the delay in the current production cycle, so it will have to speed up the task of the new production cycle in order to keep a constant production rate of 1 product each 218.268 s. All this behaviour is shown in Fig. 13 and Table 5. In Fig. 13, the first

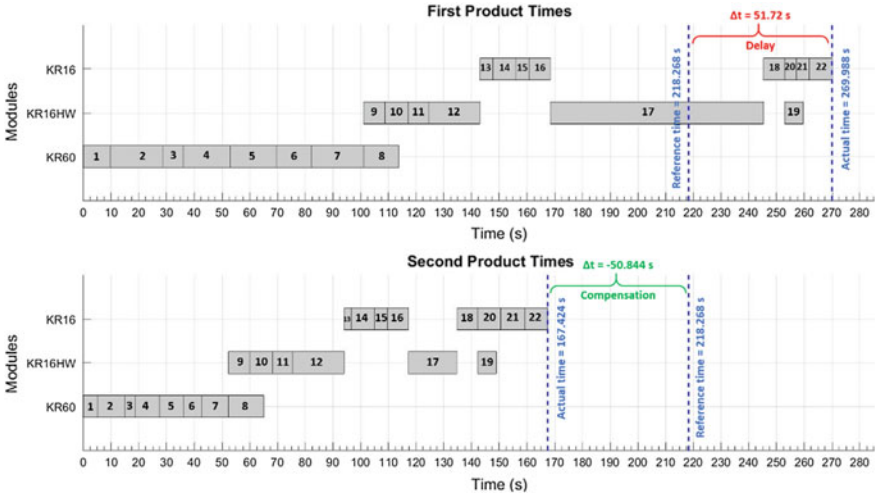

Fig. 13 Inter-production reconfiguration, the delay shown in red, the speeded-up tasks in green

Table 5 Inter-production reconfiguration times

Product 1				
Task	μT (s)	Actual time (s)	Δt (s)	Compensation (s)
KR60	101.196	101.196	0	0
KR16HW	58.932	118.728	59.796	0
KR16	58.14	50.064	59.796	−8.076
Total	218.268	269.988	59.796	−8.076
Product 2				
Task	μT (s)	Actual time (s)	Δt (s)	Compensation (s)
KR60	101.196	52.368	51.720	−48.828
KR16HW	58.932	59.148	3.108	0.216
KR16	58.14	55.908	3.324	−2.232
Total	218.268	167.424	51.720	−50.844

graphic represents the production times of the first item, where we can see that from task 1 to 16 the execution times correspond to the mean times, in the task 17 the 59.796 delay is detected, and the algorithm tries to compensate the delay in tasks 18–22. Because there are only 5 tasks left to finish the first product, the algorithm could only compensate 8.076 s, leading to a total production time for the first product of 269.988 s, this means a delay of 51.72 s with respect to the 218.268 mean production time. So, at the beginning of the next production cycle, the system will try to compensate this delay in the first tasks, as it is shown in the second graphic of the Fig. 13. It can be seen that the algorithm compensates the previous production cycle delay at task number 8. With this speed up of the manufacturing

process, the total production time of a second product is reduced to 167.424, 50.844 s below the mean production time. This way the algorithm keeps the mean production rate in 218.706 s per product, just 0.20% above the previous known production rate.

7 Conclusion

Every system is prone to failures, and manufacturing systems are no exception. The existence of failures implies delays in specific points in the manufacturing process. The main characteristic of a failure is that it can happen at any time and its duration is not defined. For all this, it is very important to have fault compensation strategy capable of monitoring at every moment if a delay has occurred in order to compensate it as fast as possible.

The proposed fault compensation algorithm together with the Distributed Control System detailed in this work, show how a delay correction can be made when a failure occurs. This correction can be achieved in 3 ways, depending on how severe the delay caused by the failure was. If the delay was short and can be compensated in a single module, a module reconfiguration takes place. Otherwise if the delay cannot be compensated by a single module, an inter-module reconfiguration is executed. And if the delay last long enough or happens in one of the last tasks of the production process, an inter-production reconfiguration is carried out.

In order to correctly implement the fault compensation algorithm, a previous knowledge of the manufacturing process is needed, to know how much time can be compensated in every task, and which tasks do not contribute to the reduction of the delay.

The experimental results of the tests carried out in the Intelligent Manufacturing Laboratory of CINVESTAV Saltillo, show the efficiency of the proposed distributed architecture and the fault compensation algorithm, implemented on a production process performed by industrial equipment such as manipulator robots, welding systems, milling tools and conveyor belts. The most important characteristic that the algorithm is capable of fulfilling is to keep the mean manufacturing rate in a production line as close as a specific known value, compensating the existence of delays due to failures or parameter changes in the working modules.

In this stage of our research, the experimental section included only delay times in the modules. However, future evaluations will be carried out for the case of a total robot failure and how another robot can re-take the task to compensate the production.

Ongoing work is currently being carried out, improving the robustness of the algorithm. Developing also a fully IoT scheme, connecting all the equipment to the cloud in order to control it from outside, capable of sharing information of the whole system with other manufacturing systems to coordinate production and transportation schedules. Work is also being developed to include safe strategies for human–robot interaction to execute task together within the manufacturing system.

References

1. Quigley M, Conley K, Gerkey B et al (2009) ROS: an open-source robot operating system. ICRA Workshop Open Source Softw 3(3):5
2. Sanfilippo F, Hatledal LI, Zhang H et al (2015) Controlling Kuka industrial robots: flexible communication interface JOpenShowVar. IEEE Robot Autom Mag 22(4):96–109
3. Shrouf F, Ordieres J, Miragliotta G (2014) Smart factories in Industry 4.0: a review of the concept and of energy management approached in production based on the internet of things paradigm. In: 2014 IEEE international conference on industrial engineering and engineering management, vol 1, Bandar Sunway. IEEE, pp 697–701
4. Wu D, Greer MJ, Rosen DW et al (2013) Cloud manufacturing: strategic vision and state-of-the-art. J Manuf Syst 32(4):564–579
5. Tang H, Huang Q, Zhang M et al (2014) Dynamic resource scheduling system with cloud. Mech Eng Autom 6:4–6
6. Yuan M, Deng K, Chaovalitwongse W et al (2017) Multi-objective optimal scheduling of reconfigurable assembly line for cloud manufacturing. Optim Methods Softw 32(3):581–593
7. Georgios A, Georgios F, Bouzakis KD (2015) Collaborative design in the era of cloud computing. Adv Eng Softw 81:66–72
8. Pei J, Liu X, Pardalos PM et al (2016) Solving a supply chain scheduling problem with non-identical job sizes and release times by applying a novel effective heuristic algorithm. Int J Syst Sci 47(4):765–776
9. Pei J, Pardalos P, Liu X et al (2015) Serial batching scheduling of deteriorating jobs in a two-stage. Eur J Oper Res 244(1):13–25
10. Yu Z, Guo F, Ouyang J et al (2016) Object-oriented petri nets and π-calculus-based modeling and analysis of reconfigurable manufacturing systems. Adv Mech Eng 8(11):1
11. Navarro-Gonzalez J, Lopez-Juarez I, Rios-Cabrera R et al (2015) On-line knowledge acquisition and enhancement in robotic assembly tasks. Robot Comput Integr Manuf 33:78–89
12. Aviles-Vinas JF, Rios-Cabrera R, Lopez-Juarez I (2016) On-line learning of welding bead geometry in industrial robots. Int J Adv Manuf Technol 83(1):217–231
13. Benitez Perez H, Lopez Juarez I, Garza Alanis PC et al (2016) Reconfiguration distributed objects in an intelligent manufacturing cell. IEEE Latin America Trans 14(1):136–146