Control of a reconfigurable assembly system

by
Azeez Olawale Adams

Thesis presented in partial fulfilment of the requirements for the degree Master of Science in Engineering at the University of Stellenbosch

Supervisor: Prof. Anton Basson
Faculty of Engineering
Department of Mechanical and Mechatronic Engineering

December 2010

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: ……………………………..

Date: ……………………………………..

Abstract

Control of a reconfigurable assembly system

AO Adams

Department of Mechanical and Mechatronic Engineering

Stellenbosch University

Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (Mechanical)

December 2010

This work considers the control of reconfigurable assembly systems using a welding assembly system as a case study. The assembly system consists of a pallet magazine, a feeding system, an inspection and removal system, a welding system and a conveyor. The aim of the work is to compare PC and PLC as controllers, as well as to compare two different approaches to reconfigurable control.

The control system of the pallet magazine was developed using a PC and a PLC. The PC control was programmed using Visual C#, while the PLC was programmed in Ladder Logic using Siemens S-300 STEP7. The two controllers were compared based on the attributes that measure the quality of a controller's software, which include its capability, availability, usability and adaptability.

The approaches to reconfigurable control considered were the agent-based methodology and the IEC 61499 distributed control methodology, both of which were applied to the feeding system. The agent-based control system was implemented using the JADE agent platform, while the IEC 61499 distributed control system was implemented using the FBDK software kit. These two methods were compared based on the characteristics of a reconfigurable system, which include the system's modularity, integrability, convertibility, diagnosability, customization and scalability.

The result obtained in comparing the PC to the PLC shows that the PLC performs better in terms of capability, availability and usability, while the PC performs better in terms of adaptability. Also, the result of the comparison between the agent-based control system and the IEC 61499 distributed control system shows that the agent-based control system performs better in terms of integrability, diagnosability and scalability, while the IEC 61499 distributed control system performs better in terms of modularity and customization. They are, however, on a par in terms of convertibility.

Uittreksel

Beheer van ‘n herkonfigureerbare monteringstelsel

AO Adams

Departement Meganiese en Megatroniese Ingenieurswese

Universiteit Stellenbosch

Privaatsak X1, 7602 Matieland, Suid-Afrika

Tesis: MScIng (Meganies)

Desember 2010

Hierdie werk beskou die beheer van herkonfigureerbare monteringstelsels met 'n sweismonteringstelsel as gevallestudie. Die monteringstelsel bestaan uit 'n paletmagasyn, 'n voerstelsel, 'n inspeksie-en- verwyderingstelsel, 'n sweisstelsel en 'n voerband. Die mikpunt van die werk is om persoonlike rekenaars (PCs) en programmeerbare-logikabeheerders (PLCs) as beheerders te vergelyk, asook om twee verskillende benaderings tot herkonfigureerbare beheer te vergelyk.

Die beheerstelsel van die paletmagasyn is ontwikkel met 'n PC en 'n PLC. Die PC-beheer is in Visual C# geprogrammeer, terwyl die PLC in leerlogika met Siemens S-300 STEP7 geprogrammeer is. Die twee beheerders is vergelyk in terme van die eienskappe wat die kwaliteit van 'n beheerder se sagteware meet en sluit in vermoë, beskikbaarheid, bruikbaarheid en aanpasbaarheid.

Die benaderings tot herkonfigureerbare beheer wat oorweeg is, is die agent-gebaseerde metodologie en die IEC 61499 verspreide-beheermetodologie. Beide is op die voerstelsel toegepas. Die agent-gebaseerde beheerstelsel is geïmplementeer met behulp van die JADE agent-platform, terwyl die IEC 61499 verspreide stelsel geïmplementeer is met behulp van die FBDK sagteware-stel. Hierdie twee metodes se vergelyking is gebaseer op die eienskappe van 'n herkonfigureerbare stelsel, waarby die stelsel se modulariteit, integreerbaarheid, diagnoseerbaarheid, pasmaakbaarheid en skaleerbaarheid ingesluit is.

Die resultate wat in die vergelyking tussen die PC en PLC verkry is, toon dat die PLC beter vaar in terme van vermoë, beskikbaarheid en bruikbaarheid, terwyl die PC beter vaar in terme van aanpasbaarheid. Die resultaat van die vergelyking tussen die agent-gebaseerde beheerstelsel en die IEC 61499 verspreide beheerstelsel wys dat die agent-gebaseerde beheerstelsel beter vaar in terme van integreerbaarheid, diagnoseerbaarheid en skaleerbaarheid, terwyl die IEC 61499 verspreide beheerstelsel beter vaar in terme van modulariteit en pasmaakbaarheid. Hulle is egter vergelykbaar in terme van omskepbaarheid.

To Her,

Mother of my Father,

Who enjoyed many years of mortal existence,

But ended it before I could finish writing this.

May Allah forgive the Lady of Malara,

And grant her admittance into the Garden of Eternal Bliss.

Amin.

Acknowledgments

I could never have completed an undertaking such as this, except with the assistance of some other people more able than myself. It therefore obviates all pretensions to self-sufficiency, and makes it necessary that I express my gratitude to these people as follows:

Many thanks to my family who I have not set eyes upon these two years past. They have remained on the sidelines cheering and urging me on. To them, I owe an incalculable debt of thanks. To this list, I must add my teacher, my friend, my brother – Siraj Obayopo and his family. He has sought out my difficulties at every turn and set to guide me aright. I also extend my gratitude to the Ibrahims. What can one say for even words fail me in expressing my profound gratitude to these two. If I had the luxury of space and the pen of a Shakespeare, then I would have indulged in their eulogy, but I am unable to and I am ashamed of my inadequacies. For this reason, I shall resort to prayers for you all.

I make special mention too of my friends – Kenny and Wasiu. How can I thank you except to hide under how guys say it in our language when they are unable to find fitting words for their expression. They say "*somo* now?" (you know now?). And Bashir and Fathi, two fine gentlemen, and the son of our Brother, Momoh Djemilou. Your companionship over these past months, I am sure, I will never forget.

Thanks too to those I shared my office and time with, and to Chris Vogt and Ruan vd Merwe, who I had the especial fortune of sharing the office with twice – guys, plenty of thanks. And if ever I had to choose an office-mate all over again, God knows – I will choose the Good German and his South African mate. In all of this, I must not forget Pieter Greff, Frank Vuureen and Frank Dymond who walked this road before us. Many thanks to them too. And to Frank Senda, the Congolese and able father of Papa Joseph.

I express my gratitude to the members of staff of the Mechanical Engineering Department of Stellenbosch University. To the secretaries on the fifth floor and others whose designation we do not know, to the kindly men of the workshop and very importantly, to Cobus Zietmann. I must remember too Tony – and his replacement Kevin Neaves who perhaps is the one amongst the whole lot I troubled most.

Thanks are also due to Stellenbosch University Library Service for their resourcefulness and efficiency – where would we be without them? And to the countless people that have assisted me in this work – who, it is very likely, I shall never again meet. To people like Johann (Distell), Francois Kleyn, Andre Liebenberg (TFD), Paul (SEW), Rushdien (Siemens) and even those who I only met in the cyber realm. Thanks too.

I would like to extend my appreciation to Advanced Manufacturing Technology Strategy (AMTS) for providing the funding for this work, and to CBI for their support. Finally, I will like to thank and make special mention of Prof AH Basson, my supervisor. I have mentioned my supervisor last and this I hasten to clarify lest some student of Language should, in future, try to infer by it my own estimation of him. Let it be known that I do this so as not to crowd him in that he is not discernible. He is one I owe much more than I can express – in my work and even more. He facilitated my coming, eased my staying and aided my completion. He gave me invaluable advice and showed such earnestness that I am unable to describe. He provided me with every necessary resource to ease my work. So far did he go – and much farther.

So, if any deficiency should be found in the quality of this work, then the fault should squarely be placed at my doorstep, not his.

Table of Contents

List of Figures

Abbreviations

CIM	Computer Integrated Manufacturing
CMS	Cellular Manufacturing System
CNC	Computer Numerically Controlled
DMS	Dedicated Manufacturing System
FMS	Flexible Manufacturing System
HMI	Human-Machine Interface
HMS	Holonic Manufacturing System
MAS	Multi-Agent System
OLE	Object Linking and Embedding
OPC	OLE for Process Control
PAC	Programmable Automation Controller
PC	Personal Computer
PLC	Programmable Logic Controller
RAS	Reconfigurable Assembly System
RMS	Reconfigurable Manufacturing System
XML	eXtensible Markup Language

1 Introduction

1.1 Background

This thesis considers the control of a reconfigurable assembly system (RAS). It is a continuation of a series of research works aimed at the development of expertise in reconfigurable assembly systems in South Africa. This research is part of the "Affordable Automation" theme of the AMTS (Advanced Manufacturing Technology Strategy). AMTS is an initiative under the Department of Science and Technology geared towards developing technologies which are related to the manufacturing industry.

The reconfigurable assembly system considered, is a welding assembly system for the components of circuit breakers manufactured by CBI (Circuit Breakers Industries) Ltd. Various students within the research group are working on different aspects of the system. The conceptual design of the welding assembly system was done by Sequira (2008). The design comprises five major systems which include a pallet magazine, a feeding system, an inspection and removal system, a conveyor and a welding system. The pallet magazine was designed by Burger (2009) and the singulation unit of the feeding system was designed by Strauss (2009). Students of Central University of Technology (CUT) are developing a multi-agent control system for the reconfigurable control of the entire welding assembly system. The multi-agent control system will interface with the controllers of each of the subsystems via OPC (OLE for process control) and will also make shop-floor data and events accessible to office managers through the internet. Jacques du Preez, a student of the Industrial Engineering Department of Stellenbosch University, is developing a simulation procedure which will determine, for a given product mix, the optimal assembly system configuration. The simulation will also predict the cost of production for the given product mix.

The work in this thesis considers the control of the subsystems designed by Burger (2009) and Strauss (2009). This will provide the control interface for the multi-agent control system to be developed by students of CUT.

1.2 Motivation

This work was motivated by the need to have an automated system which is reconfigurable. The automated system should selectively replace labour for assembly so that manual and automatic operations may be combined and run concurrently. The combination of these operations will make manufacturing more internationally competitive. The selective replacement of the workforce has become necessary due to the increases of strikes in South Africa, which can easily disrupt production plans and schedules, and the need to improve the quality of the product.

The need for a reconfigurable system stems from the fact that production volumes in South Africa are typically small, the product range is quite varied, and demand keeps changing. There is, therefore, the need to have a system that can handle a range of products and easily adapt to future changes in the type of product demanded. This cannot be achieved using a system which is not reconfigurable.

The control of the reconfigurable system is done using distributed control. The choice of a distributed method of control was motivated by the need to protect the system from the problems encountered in a centralized system of control. These problems include the high complexity of the system especially in cases of large systems, the lack of fault tolerance due to the centralized database of information and control, the high cost of maintenance of the system and the non-reconfigurability of the system. The failure of one or more components, in a centralized system of control, may lead to the malfunction or collapse of the entire assembly system. Multi-agent systems is one of the means of achieving distributed control. Furthermore, a new standard – the IEC 61499 – was recently developed as a new architecture for the development of distributed control. The standard was developed with holonic systems as one of its targeted aims. The standard hopes to make control more distributed and reconfigurable by use of function blocks. This motivated the use of this standard so that it may be compared with the multi-agent system approach.

1.3 Objective

The objective of the thesis work is to evaluate some of the current reconfigurable control strategies for some subsystems in the welding assembly cell used as case study.

The objective will be approached by, firstly, comparing PCs and PLCs as controllers. This will be done using the control of the pallet magazine as case study.

Secondly, reconfigurable control of the feeding system will be considered using both the multi-agent system approach and the distributed approach of the IEC 61499 standard. Multi-agent systems is one of the means of achieving reconfigurable control by the use of agents, while the IEC 61499 is a standard that was recently developed as a new architecture for the development of distributed control by use of function blocks. The use of these two approaches, described fully in later chapters, will enable a comparison between the IEC 61499 methodology and the multi-agent system method.

The study however does not extend to issues that arise between agents such as agent cooperation or the use of game theory in decision making. It also does not seek to find ways to optimize decision making or bargaining between agents.

2 Literature review

This chapter considers the various types of manufacturing systems that have been developed over time and the different methods available for the control of such systems. It also considers the multi-agent system method and the distributed system of the IEC 61499 standard as possible control methodologies for reconfigurable systems.

2.1 Manufacturing systems

Automated manufacturing cells are the practical building blocks of computer integrated manufacturing (CIM) systems. A cell, according to Williams (1991), can be viewed as the smallest autonomous unit capable of sustained production. The activities carried out within a cell include planning, scheduling and regulation. Planning involves generating a production plan from the process plan for the job. Scheduling involves three main tasks, which are evaluation of production, generation of a schedule containing a list of tasks with start and finish times for each equipment controller and, lastly, resolution of any conflicts and problems that may arise. Regulation includes releasing and monitoring of jobs and feedback from the equipment controllers.

Setchi and Lagos (2004) mentioned the stages of evolution of manufacturing systems from the earliest manufacturing systems. The stages they highlighted include: the Dedicated Manufacturing System (DMS), the Cellular Manufacturing System (CMS) and the Flexible Manufacturing System (FMS). As a further improvement on these stages, they made a case for the development of Reconfigurable Manufacturing Systems (RMSs).

2.1.1 Dedicated manufacturing systems

Dedicated manufacturing systems enable the production of a large number of parts on dedicated machines. The first of the manufacturing paradigms of this kind is called *mass production,* which was introduced at the beginning of the last century (Setchi and Lagos, 2004). It involved the manufacture of large product quantities with good quality at low cost. Mass production was, however, found to be wasteful of resources. As a result, a more recent paradigm called *lean manufacturing* was introduced in the 1980s (Setchi and Lagos, 2004).

Lean manufacturing aims at making more efficient use of resources. It reduces waste by producing finished products at the pace of consumer demand. Setchi and Lagos (2004) define lean manufacturing as “a systematic set of principles, methods and practices [which reduce] waste in production by reviewing all aspects of product development, manufacturing, organization, human resources and customer support.” Some of the principles of lean manufacturing they mentioned include continuous quality improvement, waste minimization and establishment of long-term relationship with customers.

2.1.2 Cellular manufacturing systems

Cellular manufacturing systems are an improvement on the dedicated manufacturing systems. CMSs consist of different cells which may be dedicated to the production of a product or a product component. These cells consist of groups of machines or workstations which are arranged in such a way that products are processed progressively without having to wait for a batch to be completed (Setchi and Lagos, 2004). A method used for the design of cells is *group technology*. Group technology, according to Setchi and Lagos (2004), is “the process of studying a large population of parts, and then grouping them into logical families with similar characteristics so that they can be produced by the same group of machines, tooling and people with only minor changes on procedure or set-up.”

An improvement to CMS is the new paradigm called *virtual cellular manufacturing,* which makes use of distributed networks and an intranet. The aim of this paradigm is to create a CMS that is more responsive to demand and changes in workload. CMSs are typically designed as groups of cells arranged physically in a particular order. It is this rigid physical arrangement that makes CMSs less responsive to changes in workload. On the other hand, in virtual cellular manufacturing, the physical cells are replaced with temporary “virtual cells”. These virtual cells are created based on a scheduling criterion. Changes in workload or scheduling criterion may lead to the creation of new virtual cells without the need for physical rearrangement, and are thus more responsive to demand and workload changes. The cells are still able to operate, in spite of these changes, because of communication over the distributed network.

2.1.3 Flexible manufacturing systems

A flexible manufacturing system is “a manufacturing system configuration with fixed hardware and fixed, but programmable, software to handle changes in work orders, production schedules, part-programs and tooling for several types of parts” (Setchi and Lagos, 2004). The main components of an FMS are computer numerically controlled (CNC) manufacturing machines, tools to operate CNC machines, robots, and automated material handling systems (Setchi and Lagos, 2004). An FMS should be “flexible” and flexibility is defined as “the ability of a system to change or react to product variation with little penalty in time, effort, cost, or performance” (De Toni and Tonchia, 1998).

There are different levels of flexibility. ElMaraghy (2006) identifies 10 types of flexibility which are as follows:

Machine flexibility: Various operations [can be] performed without set-up change,

Material handling flexibility: Various paths available for transfer of materials between machines. It can be measured by number of used paths divided by total number of possible paths between all machines,

Operation flexibility: Various operation plans available for part processing. It can be measured by the number of different processing plans available for part fabrication,

Process Flexibility: [Different] sets of part types can be produced without major set-up changes, i.e. part-mix flexibility,

Product Flexibility: Ease (in terms of time and cost) of introducing products into an existing product mix, [this] contributes to agility,

Routing Flexibility: It can be measured as the ratio of the number of feasible routes of all part types to the number of part types,

Volume Flexibility: The ability to vary production volume profitably within production capacity,

Expansion Flexibility: Ease (in terms of effort and cost) of augmenting capacity and/or capability, when needed, through physical changes to the system,

Control Program Flexibility: The ability of a system to run virtually uninterrupted (e.g. during different shifts) due to the availability of intelligent machines and system control software,

Production Flexibility: It can be measured as the number of all part types that can be produced without adding major capital equipment.

The degree of flexibility of an assembly system largely depends on its modularity. A modular design makes it easy to install, remove and regroup various modules of an assembly system. An advantage of modular design is the possibility of “plug and produce” (Martinsen et al, 2007). This means that modules can be dynamically added or removed from the system without having to change or reconfigure the hardware or software of the assembly system.

Many factors have been identified that militate against the adoption of FMSs and have even prompted the need for the development of RMSs. Raj et al (2007) discussed some of the issues surrounding the implementation of FMSs. They grouped the issues identified into seven classes as issues regarding parts loading, scheduling, material handling, flexibility, machine tools, operation and control, and the human element. They, however, concluded that there is no definite recommendation on the procedure for the implementation of FMSs. Apart from the issues of implementation mentioned above, there are also some barriers that inhibit the transition of firms from other manufacturing systems to FMSs. Raj et al (2007) identified these barriers as the high cost and uncertainty of FMSs, the problem of tool management, the difficulty of design, and flexibility since most FMSs exhibit volume flexibility but lack product flexibility. They, however, did not propose solutions to the individual problems identified, but highlighted the importance of knowledge of FMSs before trying to implement such a system.

Mehrabi et al (2002), on the other hand, conducted a survey of experts in manufacturing in the industry on the use and adoption of FMSs and RMSs. The result of the survey shows that "it appears that FMSs have excess capacity and features which in many cases were not eventually used. Furthermore, their complexity, high initial costs, lack of reliability of the software, the needs for highly skilled personnel and support costs, and lack of capability and willingness of machine tool builders to carry out necessary system engineering involved are among the reasons that make FMSs not very attractive to industry" (Mehrabi et al, 2002).

2.1.4 Reconfigurable manufacturing systems

An RMS is a system designed for rapid change in structure in order to quickly adjust production capacity and functionality within a part family in response to changes in market requirements. The objective is to provide exactly the functionality and capacity that is needed, when it is needed (Koren et al, 1999). The idea of reconfigurability is consistent with that of expansion flexibility (ElMaraghy, 2006). For this reason, there are a number of similarities between flexible systems and reconfigurable systems which have sometimes made it difficult to distinguish between the two systems.

Bi et al (2008) point out the controversy surrounding the definition of RMSs. They mention, as an example, the 3rd Conference on Reconfigurable Manufacturing held at the University of Michigan during May 10–12, 2005, where "some [people] insisted that an RMS is an intermediate paradigm between Mass Production and Flexible Manufacturing Systems (FMSs), some argued that an RMS is an advanced paradigm whose flexibility must be higher than that of an FMS, and others said it is not very meaningful to distinguish RMSs from FMSs." The authors, however, concluded that an RMS is a system which has the "ability to reconfigure hardware and control resources at all of the functional and organizational levels, in order to quickly adjust production capacity and functionality in response to sudden changes in market or in regulatory requirements."

According to Wiendahl (2007), ***reconfigurability*** describes the operative ability of a manufacturing or assembly system to switch with minimal effort and delay to a particular family of work pieces or sub-assemblies through the addition or removal of functional elements, while ***flexibility*** refers to the tactical ability of an entire production and logistics area to switch with reasonably little time and effort to new – although similar – families of components by changing manufacturing processes, material flows and logistical functions.

Key to the difference between reconfigurability and flexibility are:

- The diversity of workpieces handled. Reconfigurable systems may switch between *different families* of products, while flexible systems switch between *similar* products

- The extent of change the manufacturing system has to undergo. Reconfigurable systems may *add or remove* machine components, while flexible systems *change* the process or material flow.

Apart from flexibility, Koren et al (1999) mention five important characteristics of RMSs. ElMaraghy (2006) summarizes these and adds an additional characteristic. These were given as:

i. ***Modularity*** of both hardware and software components,

ii. ***Integrability*** for both ready integration and future introduction of new technology,

iii. ***Convertibility*** to allow quick changeover between products and quick system adaptability for future products,

iv. ***Diagnosability*** to identify quickly the sources of quality and reliability problems,

v. ***Customization*** to match designed system capability and flexibility to applications,

vi. ***Scalability*** to incrementally change capacity rapidly and economically (Elmaraghy, 2006).

There are two types of reconfiguration that can occur in a manufacturing system. Rooker et al (2007) mentions these as basic and dynamic reconfiguration. Basic reconfiguration is reconfiguration in its simplest form, which can be achieved by stopping the system, applying the necessary hardware changes, and then restarting the system. This is also called “coldstarting” the system. Dynamic reconfiguration, on the other hand, is reconfiguration which takes place while a system is still in operation without having to stop the system. Timeliness is the crucial factor in dynamic configuration (Rooker et al, 2007).

Bi et al (2007a) mention some of the issues involved in the development of RMSs:

- Separation of RMSs from product design. As currently designed, most RMSs are developed separate from the product design and this makes optimization of the system difficult
- Perception of RMSs as a premature technology. RMSs are still in their early days and full automation cannot yet be achieved, so developers still have to solve many of the problems manually
- Indifferent attitude towards RMSs. The attitude towards RMSs is not encouraging. A number of companies are uncertain of the importance of automating their assembly systems
- Use of an RMS as a wrong solution. RMSs do not have to be deployed in all manufacturing situations. There are some cases where RMSs may not be the most suitable solution, especially where there is a lack of technical competence or where the company has no need to adapt to different manufacturing strategies.

2.1.5 Holonic manufacturing systems

The quest for decentralization gave rise to the concept of holonic manufacturing systems. The term holon, was first introduced in 1967 by Koestler (Paolucci and Sacile, 2005) from the Greek word “holos” which means “whole”. A holon, as Koestler named the term, is a part of a (manufacturing) system that may be made up of subordinate parts, and in turn, can be part of a larger whole (Leitao and Restivo, 2008). This concept is used to refer to the decentralized coordination and control of manufacturing systems, hence the term holonic manufacturing system. The concept of holon and holon system has a wide and varied application. Ermolayev and Matzke (2007) mention many applications of HMSs, including emergency response, e-commerce, traffic control, engineering design and, of course, agile manufacturing.

Implementation of this manufacturing system can be done by the use of an agent-based control system. In fact, Ermolayev and Matzke (2007) state that software agents and agency paradigms are a "natural choice" for modeling holonic systems. Agents have been used in many different fields of human endeavor but its application to manufacturing systems is quite recent. Because of the dearth of defined ways of applying this method to manufacturing systems, Bussman et al (2004) developed a methodology to help the control engineer apply the use of agents to manufacturing control. This methodology was termed the DACS (Designing Agent-based Control Systems) methodology.

2.2 Control of manufacturing systems

2.2.1 Types of control architectures

Meng et al (2006) mention three types of control architectures: centralized, hierarchical and distributed. As illustrated in figure 2.1, centralized control is one in which the entire system is controlled by one controlling system which carries out all the automation processes. Hierarchical control generally involves more than one controller arranged in some form of hierarchy. In hierarchical control, control decisions emanate from the highest level of the hierarchy, and these are then decomposed into smaller and more detailed instructions which are passed on to the lower level controllers for implementation. Distributed control, on the other hand, involves independent control and handling of the various automation processes by different controllers which interact with one another by engaging in some form of communication.

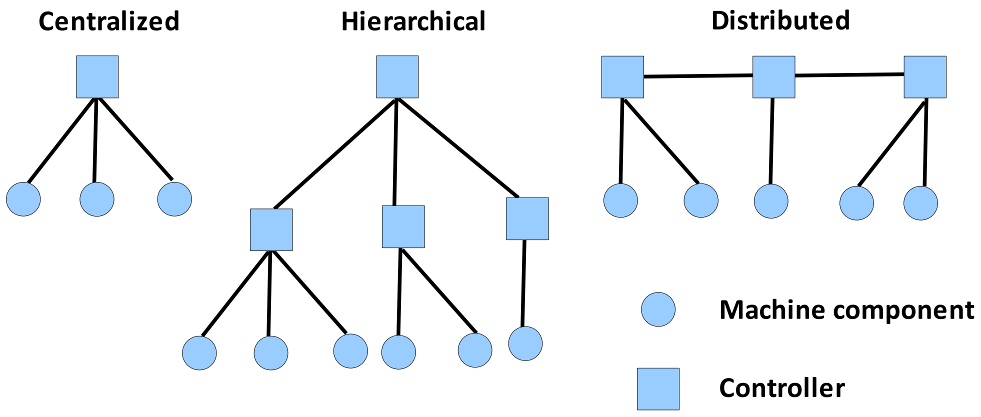


Figure 2.1 Three types of control architectures (Meng et al, 2006)

The classical approach to control of production systems is hierarchical and schedule-driven. However, there has been a shift away from the hierarchical approach to control and the recent trend has been towards the implementation of control methods which are more heterarchical and distributed. An example of a system with a heterarchical structure, in which control is distributed, is the holonic manufacturing system (Scholz and Freitag, 2007).

Scholz and Freitag (2007) stated some of the disadvantages of hierarchical control and the advantages of heterarchical control over them. They mentioned that the disadvantage of hierarchical control is largely due to the complexity of these control systems which grows rapidly with the size of the manufacturing system. This complexity results in high costs for development, maintenance, operation, and modification of the control system. On the other hand, the advantages of heterarchical control are that:

- it leads to reduced complexity by localizing information and control,
- it reduces software development costs by eliminating supervisory [control] levels,
- it has higher maintainability and modifiability due to improved modularity and self-configurability,
- it has improved reliability by taking a fault-tolerant approach rather than a fault-free approach (Scholz and Freitag, 2007).

Centralized control has a number of shortcomings. Meng et al (2006) give some of these shortcomings as structural rigidity, difficulty of control system design, lack of flexibility and a low level of fault tolerance. It is difficult to add, modify, or delete resources. In order to reconfigure a centralized or hierarchical system, the system has to be shut down and all data structures of higher levels need to be updated. Unforeseen disturbances, such as machine breakdown, invalidate the production plan and schedule. Because of these, centralized and hierarchical modes of control are not suitable for RMSs (Meng et al, 2006).

Different methods have been used to implement control (either hierarchical or heterarchical) in different manufacturing systems. Some of these methods are specific to certain manufacturing systems, while others are more general. Below, we discuss the various methods that have been applied to the different manufacturing systems. The agent-based control method and the distributed control method based on the IEC 61499 standard are methods that can be specifically applied to RMSs and HMSs respectively. The agent-based method will be discussed in section 2.3, while the IEC 61499 methodology will be discussed in section 2.4.

2.2.2 Control method for FMSs

Ferrolho and Crisostomo (2007) worked on control and integration software for FMSs. They developed customized software for different equipment that make up the components of an FMS, and an integration software unit which allows an easy and efficient integration of these components into an FMS. Each customized software unit makes use of the original control capacity of the equipment it was developed for by acting as a client, while the equipment's control system serves as a server. Examples of the software developed include the two software programs named “winRS232ROBOTcontrol” and “winEthernetROBOTcontrol”, which were developed for different industrial robots depending on whether they communicate via RS232 or Ethernet. They also developed software programs named “winMILLcontrol” and “winTURNcontrol” for the CNC mill and CNC lathe respectively. They tested the software in an industrial application and concluded that the software is viable and that it resulted in improved performance of the FMS.

2.2.3 Control methods for RMSs

Software issues represent the area of greatest concern for the successful development of the RMS technology (Mehrabi et al, 2002). For this reason, there has been widespread research into different methods of controlling RMSs. Some of the methods used so far include Petri nets, HMI based control and multi-agent systems. The control of a reconfigurable system is similar to that of a distributed manufacturing system (Bi et al, 2007b).

Petri nets are used as one of the methods of controlling reconfigurable systems. A Petri net is a graphical representation of discrete event systems. It is a directed graph consisting of *nodes* indicating transitions or events, *places* indicating conditions and *directed arcs* indicating relations between events. The nodes, places and directed arcs are represented using bars, circles and arrows respectively. The method was invented by Carl Petri in 1939 to describe chemical processes. An example of a Petri net model is shown in figure 2.2. The system consists of two related systems or automata. The states or places in the first automaton are shown as the circles marked *s1*, *s3*, *s4*, *s6*, while the states of the second automaton are the circles *s2*, *s5*, *s7*. The transitions are the squares marked *t1, t2, t3, t4, t5, t6, t7* and they show the events which can cause an automaton to move from one state to the other. From the diagram, it can be seen that the automata synchronize on the transitions *t4* and *t5*.

Petri nets also serve as a powerful tool for modeling and control of assembly systems by considering them as discrete event systems. This method was used by

Yu et al (2003) and Kuo et al (1999) as a tool to model and control assembly systems.

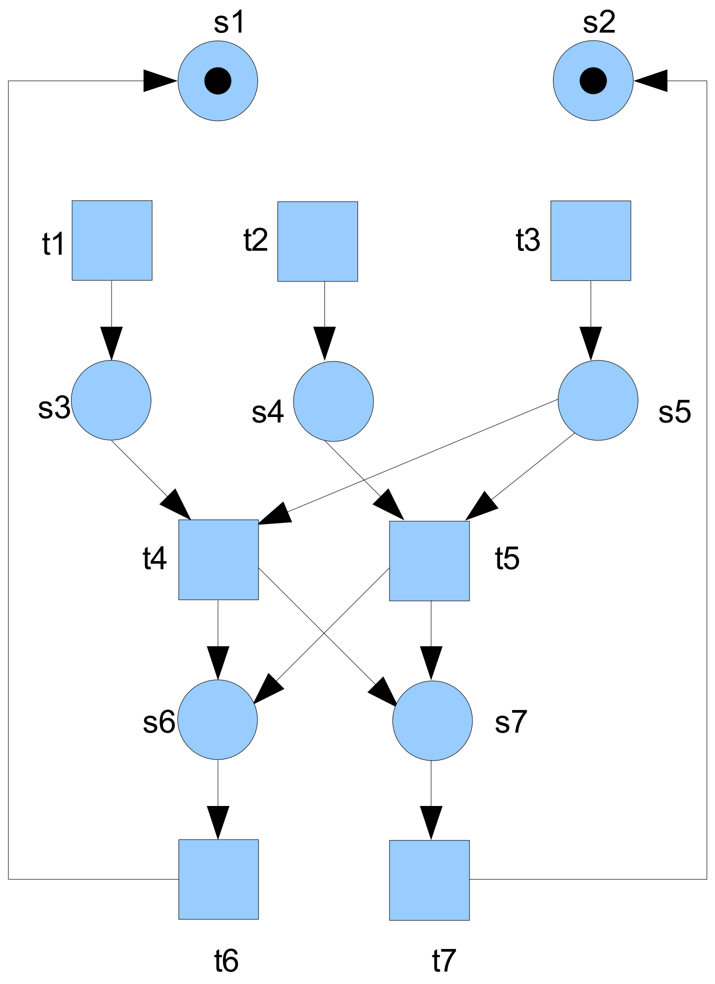


Figure 2.2 Petri net model (Partial-order Verification Techniques, 2004)

Although Petri nets have been used to a large extent to model manufacturing systems, the classical Petri net which can be used to describe logical control lack the ability to treat information flow. They are not data-oriented (Yu et al, 2003). This motivated the introduction of artificial intelligence into the use of Petri nets as done by Yu et al (2005). They discussed the strategy of modeling RMSs using a method called Knowledge Based Timed Colored Object-oriented Petri Net (KTCOPN). KTCOPN is the result of the combination of knowledge and object-oriented methods with timed colored Petri net. Using KTCOPN, the modeling of RMSs was done in three phases: the construction of the object-oriented Petri net (OPN) for the assembly cell, the construction of the OPN for the assembly module and the construction of the entire KTCOPN model for the reconfigurable assembly system. The modularity introduced into the system ensures the quick reconfiguration of the system (Yu et al, 2005).

Onofrio and Bruccoleri (2006) developed an HMI-based control system for a reconfigurable manufacturing cell. The control system was developed using object-oriented methodology and was programmed with Microsoft Visual Basic. The system has an interactive user interface and it allows for easy reconfiguration by reacting to changes in both the operations sequence of workpieces and in the hardware configuration of the manufacturing cell (Onofrio and Bruccoleri, 2006).

Agent-based control is another method that may be applied in the control of RMSs, but this will be discussed in section 2.3.

2.2.4 Control methods for HMSs

After the conception of the idea of HMSs, an international consortium was formed. This consortium on the HMS was set up as one of the projects under the Intelligent Manufacturing Systems (IMS) program. Its aim was to standardize, research and create support for the HMS architecture by covering such topics as "system architecture and engineering, planning and scheduling, control and holonic man–machine system and emulation" (Kotak et al, 2003). Some of the research outputs of this consortium include:

- the development of a holonic system architecture by Van Brussel et al (1998) called PROSA (Product-Resource-Order-Staff Architecture)
- the development of a methodology and architecture for holonic multi-cell control system by Langer (1999) as part of his PhD thesis
- the presentation of an architecture for the coordination of a holonic automated guided vehicle system by Liu et al (2000)
- the development of a holonic production planning and control system by McFarlane and Bussmann (2000)
- the development of a virtual manufacturing environment to implement the holonic shop floor control by Kotak et al (2003).

In addition to the work done by the consortium, other researchers have also developed different control architectures for HMSs. These include ADACOR (***ADA***ptive holonic ***CO***ntrol a***R***chitecture), MetaMorph I and II, HCBA (Holonic Component Based Architecture), RFID based approach, etc.

ADACOR, which is also called a collaborative control architecture, was developed by Leitao (Leitao and Restivo, 2006). It is built on a set of cooperative holons which are used to represent different manufacturing components. These components may be physical entities, for example machines, pallets, etc, or logical entities such as products and orders within the system. Four holons relating to manufacturing are specified in ADACOR and they include the product holon (PH), the task holon (TH), the operational holon (OH), and the supervisor holon (SH) (Leitao and Restivo, 2006). The PHs, THs, and OHs are holons which represent the different products, orders and resources within the manufacturing system respectively. The SH is the holon responsible for the coordination and optimization of the other holons. The holons in ADACOR are implemented as agent classes in the JADE (Java Agent DEvelopment) framework, which is a software unit that can be used to develop agents. JADE complies with Foundation of Intelligent Physical Agents (FIPA) specifications for agents and communication between the autonomous holons can be done using FIPA agent communication language (ACL).

The control architecture described in PROSA is similar to ADACOR. PROSA specifies three main holon classes: the resource holon, the product holon and the order holon (Van Brussel et al, 1998). These holon classes are referred to as the basic holons and that is because they are present in every HMS. The resource holon is used to represent physical entities such as machines, while the product holon is used to represent the process and knowledge about the product that facilitates product processing. The order holon represents a task that is to be done in the manufacturing system. Apart from the basic holons, a staff holon is also defined in PROSA. The staff holon is a kind of “utility” holon which assists the basic holons in performing their functions. The function of the staff holon in PROSA is not restricted to supervision as in the case of the supervisor holon in ADACOR.

Another holonic control architecture is MetaMorph which was developed at the University of Calgary. MetaMorph consists of two approaches: MetaMorph I (which is now called MetaMorphic) and MetaMorph II. MetaMorph I consists of resource agents and mediator agents. The model used is that of a hybrid agent model. The mediator agents act as brokers and recruiters for the resource agents (Shen et al, 2000). They act as brokers by receiving information or requests from an initiating agent. They interpret the information or request and then look for a receptor agent. They also act as recruiters by searching for agents based on the criteria they receive from one of the resource agents. They link the agents that match the given criteria with the requesting agent so they can communicate with each other. The MetaMorph II approach is an extension of MetaMorph I. It involves not just the integration of distributed intelligent machines, but also the integration of other aspects of manufacturing, such as planning, scheduling, execution, material supply, market services, etc, into a distributed intelligent open environment (Shen et al, 2000). Though the implementation is similar to MetaMorph I, in MetaMorph II, there are additional mediator agents such as Shop Floor Resource Mediators, Machine Mediators, Tool Mediators, Worker Mediators, etc.

Chirn and Farlane (2000a) introduced the component-based approach to holonic systems. They applied this approach to the control of a robot assembly cell. The idea originated from the concept of Software Integrated Circuit (SIC). Their aim was to develop and package software components for later use in the same way as hardware components are packaged for later use in integrated circuits (Chirn and Farlane, 2000b). "The Component Based Development (CBD) approach focuses much on developing reusability and reconfigurability in view of the architecture rather than the individual software modules" (Chirn and Farlane, 2000a). They introduced two holons: product and resource holons. The resource holons are independent and are not allowed to communicate directly with each other. This is to avoid poor integration when dealing with long-term changes, and to ensure easy replacement and reconfiguration of the system. The product holons, on the other hand, make use of the resource holons by negotiating with them.

Generally, the implementation of the control architectures of HMSs is done using multi-agent systems. This is not particularly surprising because agents and holons share many common attributes which include being autonomous, cooperative and open (Kotak et al, 2003). Bussmann and Sieverding (2001) undertook an industrial evaluation of the holonic manufacturing system which was implemented using multi-agent systems. They concluded that the holonic paradigm does meet the requirements of an industrial deployment, increasing scalability and productivity of the assembly process, while maintaining high volume and low costs per product.

Kamioka et al (2007) developed an RFID-driven holonic control scheme for production systems. In this scheme, an RFID tag is attached to each product component (which is independent and is considered to be a holon). This RFID tag contains the "lifeline" information necessary for the processing of each component. Based on the information on the RFID tag on each component, the controllers of the conveyors are able to direct the component to the relevant production facility which will process it. When the processing is complete, this facility returns the processed component to the next conveyor which takes it further on to the next relevant production facility based on the RFID tag information. Lastly, in case of a change of orders (i.e. reconfiguration of the system), the new and updated information is sent to all the relevant production facility controllers. Each controller compares the information on the product's RFID tag with the order change information and decides whether the component should be processed or not. If the component is to be canceled, the cancellation-related information including the ID for its destined inventory center is written on the RFID tag. The canceled component is then transported to the relevant facility by the conveyors.

The use of RFID tags in holonic control offers a great advantage as regards flexibility. This is because Gouyon et al (2007) state that having RFID tags embedded on products enables *individual* identification of product occurrences. This individual identification opens a way towards the customization of control rules for each product occurrence, which implies greater flexibility (Gouyon et al, 2007). However, the use of RFID tag information *alone* in a control scheme, as done by Kamioka et al (2007), is not a reliable method of control. Gouyon et al (2007) state that the reliability of read and write operations on RFID tags is not yet 100%. Therefore, a control method that is entirely dependent on information on RFID tags cannot be accurate. It has to be coupled with other forms of control.

Another methodology for the control of HMSs is by the use of the distributed control based on the IEC 61499 standard. This method will be discussed in section 2.4.

2.3 Agent-based control

Agent-based control is achieved by agents in a multi-agent system. A major component of using agent-based computing to solve a problem is the decomposition of the problem into various autonomous entities which solve the problem. Decomposing the problem simplifies complex systems in two ways:

- Firstly, it gives a natural representation for complex systems that are invariably distributed, which is a suitable condition in the case of reconfigurable assembly systems.
- Secondly, due to the devolution of actions to autonomous entities, the actions performed by these entities (or agents) can be said to be responsive to the agent's actual state of affairs, rather than some external entity's perception of this state (Jennings, 1999).

2.3.1 Agents and agent communication

Agents have been defined by many authors, but a well suited definition for our application is the definition given by Jennings and Wooldridge (1998), which has often been cited by other authors: “An agent is considered [to be] a software entity situated in a production environment, with enough intelligence that is capable of autonomous control actions in this environment and of co-operation relationships by participating in associations with other entities in order to meet its design objectives". An agent should be able to act without the direct intervention of humans or other agents, and should have control over its own actions and internal state (Jennings and Wooldridge, 1998).

Multi-agent systems are made up of interacting agents and these agents possess a number of properties which make them intelligent. Agents could be autonomous, proactive, cognitive, adaptive or reactive (Guessoum, 2004). The most important of all of these properties is autonomy in decision-making (Tozicka et al, 2007). Another key property is reflectivity which is the ability of agents to observe and understand their behavior, reason about their behavior and revise the behavior accordingly (Tozicka et al, 2007). Most agents have one or more of these properties. The agents in multi-agent systems must be aware of their own capabilities and of changes to other agents and their environment. To remain effective, agents must be able to adapt their structures and knowledge while they execute (Guessoum, 2004).

There are two dominant approaches to the way agents are modeled: the cognitive approach and the reactive approach (Guessoum, 2004). In the *cognitive* approach, each agent is a symbolic model of the real world in which it is supposed to operate. Based on the information it possesses of its environment, it develops plans or makes decisions using the traditional methods of artificial intelligence. On the other hand, in the *reactive* approach, "simple-minded agents react rapidly to asynchronous events without using complex reasoning" (Guessoum, 2004). Reactive agents are behaviour-based agents. These agents are defined simply by a set of behaviours which determine their reaction to events, and therefore, they do not need to have memory (Tang and Wong, 2005). A third approach to agent modeling is the hybrid approach as mentioned by Bussmann et al (2004). They suggested three agent architectures: reactive agents (based on the reactive approach), deliberative agents (based on the cognitive approach) and hybrid agents (based on the hybrid approach). The hybrid agents incorporate both reactive and deliberative mechanisms in one architecture (Bussmann et al, 2004).

While agents may be able to determine their individual plans based on their own competencies and knowledge, there is the need for agents to interact with other agents, by communicating and sharing information, in order to solve complex problems and avoid conflicts. The need for interaction between agents necessitated the development of standards for agent development and communication. This gave rise to the Foundation for Intelligent Physical Agents

(FIPA). “FIPA is an IEEE Computer Society standards organization that promotes agent-based technology and the interoperability of its standards with other technologies” (FIPA, 2010). FIPA was founded in 1996 as an organization of academic and industrial organizations, but officially became an IEEE standards organization in June 2005.

The core of the FIPA standards is the agent communication standard (FIPA, 2010). Agent communication in the FIPA standard is according to the Agent Communication Language (ACL) and is called the FIPA-ACL. The FIPA-ACL defines standard acts such as INFORM, AGREE, REQUEST, etc, which agents require in communicating with each other. These acts are based on the speech act theory. This theory assumes that messages represent actions or communicative acts, also known as speech acts or performatives (Bellifemine et al, 2004). The first ACL was the Knowledge Query and Manipulation Language (KQML) that included many performatives, assertives and directives which agents use for telling facts, asking queries, subscribing to services and/or finding other agents (Monostori et al, 2006).

The FIPA standard, however, does not specify any particular “language” to be used along with the acts specified in the standard. Any language or representation may be used, but FIPA has its own FIPA-SL language which is widely recommended. Some of the standards also specified by FIPA include Agent Management System (AMS), the Directory Facilitator (DF), the Agent Platform, etc.

In multi-agent systems, agents need to engage in some form of bargaining or negotiation in order to reach certain decisions. For this reason, many models for negotiations have been developed. For example, Turgay (2008) proposed some decision-making rules for the control of agent-based manufacturing systems, while Qiu et al (2004) applied non-cooperative game theory as a means of facilitating the decision-making process during reconfiguration at the machine controller level. Zhao et al (2008) used an optimization model called the particle swarm optimization model to achieve dynamic reconfiguration and task allocation within a multi-agent system. Zhao et al (2008) also made use of the contract net protocol for agent interaction. Another method is the use of auctions which Mahr and de Weerdt (2007) declared to be faster and more efficient than bargaining.

2.3.2 Use of agents in manufacturing control

The multi-agent system approach is a convenient and well-tested approach to reconfigurable control. It is one of the most adopted technologies used in RMS and HMS paradigms’ applications (Candido and Barata, 2007). This is because multi-agent systems increase the “plug and produce” capability of the manufacturing system by making it possible for components to enter or leave the system with minor variations in the production process. This is enabled by components' modularization and embedded intelligence, which together ensure close to zero-downtime reconfigurability (Candido and Barata, 2007). In fact, Turgay (2008) made a review of the different methods that have been applied to the control of RMSs and concluded that "multi-agent systems (MASs) offer modularity. If a problem domain is particularly complex, large or unpredictable, then the *only* way it can *reasonably* be addressed is to develop a number of functionally specific modular components (agents) that are specialized in solving a particular problem aspect".

The use of agents in the control of RMSs was implemented by Sugi et al (2003), Tang and Wong (2005), and Wang et al (2005). The agent-based control architecture was also used to develop the NovaFlex Shop Floor Environment (Candido and Barata, 2007). Farlane et al (2001) developed an algorithm for agent-based control of manufacturing flow shops in which they utilized the queuing theory. Lohse et al (2005) developed an ontology based agent control system. An ontology is a set of concepts and symbols used to express messages sent between agents (Bellifemine et al, 2004). Lohse et al (2005) used an ontology which defines the assembly system requirements in terms of product and assembly process descriptions, and the capabilities of assembly requirement modules in terms of the equipment functions, behavior and structure.

Al-Safi and Vyatkin (2007) discussed the use of a reconfiguration agent which can be used for reconfiguration without human intervention. The reconfiguration agent uses its ontological knowledge of the manufacturing environment for reconfiguration. It attempts to reconfigure the system whenever it realizes that the current configuration is not able to fulfill the required task whether due to changes in the manufacturing requirement or the manufacturing environment. The use of the agent minimizes the overhead of the reconfiguration process and achieves rapid reconfiguration (Al-Safi and Vyatkin, 2007).

Ulieru (1997) analyzed the design of control mechanisms for a multi-agent flexible transfer system. The system consists mainly of two groups of agents: the specialist agents and the supervisor agents. The supervisor agents represent pallets which have workpieces on them, while the specialist agents represent the various machines which work on the workpieces. The supervisor agents are autonomous and are able to chart the path to follow in order to accomplish the work to be done on the workpiece, while the specialist agents have cognitive capability and are able to respond to several unexpected situations.

It has often been thought that the object-oriented methodology can also be applied to the control of reconfigurable systems. The two well-known software engineering technologies (multi-agent systems and object-oriented software engineering) seem well suited to implement a holonic abstraction of a reconfigurable control problem. This is because multi-agent systems have a distributed nature and object-oriented systems have a recursive structure [i.e. a hierarchical structure in which complex elements could be created from simpler ones through inheritance] (Colombo et al, 2006). However, Bussmann and Jennings (2003) compared the two software techniques and concluded that the use of agents is more suited for complex problems. They state the following in support of the use of agents over objects:

- Objects are generally passive in nature. They need to be sent a message before they become active
- Although objects encapsulate state and behavior realization, they do not encapsulate behavior activation (i.e. action choice). Thus, any object can invoke any publicly accessible method on any other object. Once the method is invoked, the corresponding actions are performed
- Object orientation fails to provide an adequate set of concepts and mechanisms for modeling complex systems. Recognition of this fact led to the development of more powerful abstraction mechanisms such as design patterns, application frameworks, and component-ware. Although these are a step forward, they [still] fall short of the complete set of data desired for the development of complex systems. They focus on generic system functions, and the mandated patterns of interaction are rigid and predetermined
- Object-oriented approaches provide only minimal support for specifying and managing organizational relationships.

Agents are also suitable for control of HMSs because agent technology provides techniques for modeling and implementing autonomous and cooperative software systems. Agents can even be viewed as holons without physical processing capabilities (Bussman and Sieverding, 2001). Kotak et al (2003) present the senario of an agent system used in the control of a HMS as shown in figure 2.3. Each holon is represented by an agent in the system, and together with other holons, they form a holarchy. Each holon has its agent (software) aspect and physical aspects comprising the device and drivers. The Directory Facilitator, which is platform-dependent, stands in to coordinate the other agents.

However, there are some downsides to the application of the agent-based approach to engineering problems. These limitations include:

- The patterns and outcomes of the interactions between agents are inherently unpredictable
- Predicting the behaviour of the overall system based on its constituent components is extremely difficult (and sometimes impossible) because of the strong possibility of [unintended] emergent behaviour (Jennings, 1999).

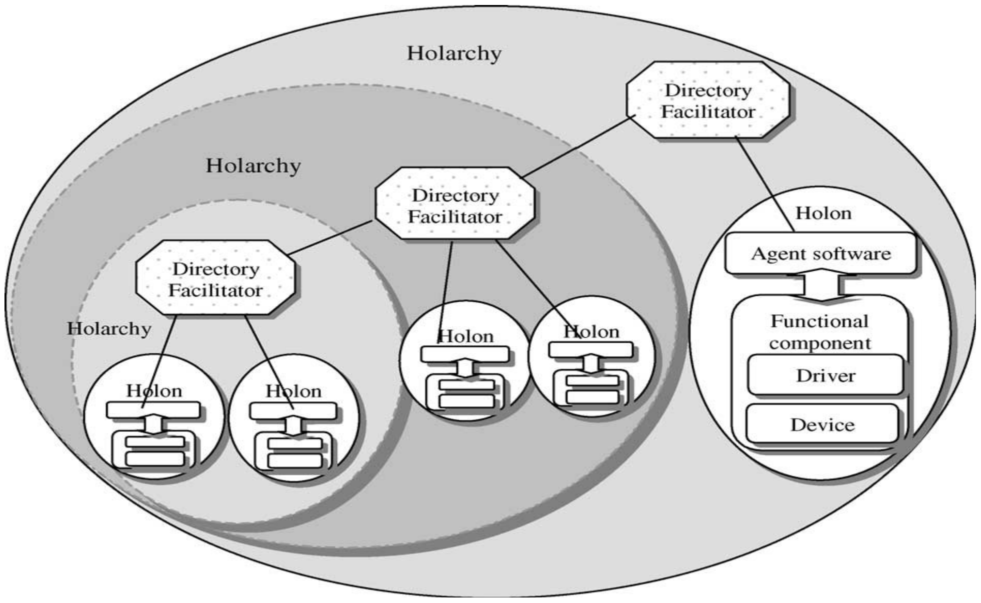


Figure 2.3 Agents used to represent holons in HMS (Kotak et al, 2003)

2.3.3 Methodologies for developing agents

Different methods of developing multi-agent systems exist. Some of the methods are general, while some are more specific methodologies for creating particular categories of agents. Some others have been developed based on the agent platform on which the agent system will be implemented. Bussmann et al (2004) gives an extensive list of agent methodologies and their categories. Included in the list are knowledge-oriented methodologies, e.g. CoMoMAS, MAS-CommonKADs, etc, some methodologies for manufacturing, e.g. PROSA, some role-based methodologies, e.g. MASB and Gaia, and some system-oriented methodologies, e.g. MESSAGE and Prometheus, etc.

Bussmann et al (2004) developed a methodology called DACS (Designing Agent-based Control Systems) specifically for manufacturing control. The DACS methodology consists of three main steps: analysis of control decisions, identification of agents and selection of interaction protocols. Analysis of control decisions, which is the first step, includes the identification of effectoric decisions and the identification of decision dependencies. Effectoric decisions are decisions which result in at least one physical action by the machines, while decision dependencies are the relationships that exist between effectoric decisions. The outcome of the first step is called the decision model. The second step, which is the identification of agents, involves clustering of decision tasks and improving the decision model. The result of this step is a list of agents. The final step is a selection of interaction protocols. This results in the agent-based design.

Agents can be simulated on traditional object-oriented programming languages, but this is error-prone and fraught with as much difficulties as attempting to develop objects using a non-object-oriented language (Padgham, 2004). Various software on which agent systems can run have been developed. Most of the software are middleware platforms which are ported on JAVA. The agent software platforms allow programmers to write agent programs using their knowledge of the JAVA programming language, while the software takes care of such details as "agentisation" (i.e. building the agents from the code), agent communication, etc. Just as in the case of agent methodologies, some agent-oriented software are general, while others are specifically for particular types of agents. Padgham and Winikoff (2004) classified agent platforms into three groups based on the "strengths" of the platforms. These are:

- Agent platforms that support internal agent reasoning and the development of agent plans, goals, etc. Examples of these platforms include PRS, JACK, JADEX, etc.
- Agent platforms that focus on inter-agent communication and provide means for transfer of messages between agents. Examples of these platforms include JADE, Zeus, etc.
- Agent platforms that focus on agent mobility. Examples of these platforms include Grasshopper, Aglets, etc (Padgham and Winikoff, 2004).

Examples of other agent platforms that have been developed for specific types of agents are the GOAL Agent Programming Language for developing "rational agents" and the 3APL for developing cognitive agents.

Rzevski et al (2007) developed a toolkit called the Magenta Toolkit which is a set of multi-agent tools for developing large-scale adaptive multi-agent applications. This toolkit has found wide application in, for example, the collaborative design of an airplane wing, web portal for healthy lifestyle, and ocean and truck schedulers (Rzevski et al, 2007).

2.4 Distributed control based on IEC 61499

Industrial personal computers (PCs) and programmable logic controllers (PLCs) are the major controllers used for automation. Control application in PLCs have been based on the IEC 61131-3 standard, but in 2005, the IEC 61499 standard, which had been in the works for some time, was officially released to extend the IEC 61131-3 standard.

The IEC 61131 (formerly IEC 1131) standard was set up to provide a global standard for PLCs ranging from PLC programming to PLC communication via fieldbus. The standard consists of several parts with each part dealing with specific aspects relating to PLCs. The part of the standard that deals with PLC programming is the IEC 61131-3. One of the major objectives of the IEC 61131-3 standard is to improve the quality of PLCs with regards to:

- "*Capability*", which is the extent to which a system can perform its intended design functions
- *"Availability"*, which is the proportion of time in the life of a system when it is available for its intended design functions
- "*Usability*", which is the ease with which a specified set of users can acquire and exercise the ability to interact with the system in order to perform its intended design functions
- *"Adaptability"*, which is the ease with which a system may be changed in various ways from its initial intended design functions (Lewis, 1998).

In spite of the effort put into the development of the standard, some of these goals were not met. An example is "Portability" which Lewis (1998) mentions as one of the factors influencing *Adaptability*. It is generally known that not all PLC programs are portable from one PLC manufacturer to the other. In order to promote wide acceptance of the IEC 61131-3 standard, and PLC interoperability, the PLCopen organization was founded. PLCopen was set up by PLC manufacturers, and is a vendor-independent association which ensures various levels of compliance with the standard.

The deficiencies in the current PLC software influenced studies by various research groups and highlighted the need for a new standard, called IEC 61499, different from IEC 61131-3. According to Zoitl et al (2007), two of these studies had the most far reaching effect in the development of the IEC 61499 standard. The first study, by the Iacocca Institute, developed the concept of "agile manufacturing" which envisaged not only a dynamic reconfiguration of control applications, but also a physical reconfiguration of production resources (Zoitl et al, 2007). The other study, which was carried out by the HMS project consortium, developed the "means and methods for self-adaptable production systems", which led to the HMS with a new lower level control architecture (Zoitl et al, 2007). These major researches influenced the extension of the function blocks of

the already established IEC 61131-3 standard to include event-driven execution in the new IEC 61499 architecture.
The drawbacks in IEC 61131-3 standard which have been addressed in IEC 61499 are:

- non-deterministic switching points in time (due to cyclic execution policy),
- lack of fine granularity (i.e. reconfiguration at task level),
- jitter effects (i.e. task reconfiguration influences other tasks),
- the possibility of inconsistent states (which may lead to deadlocks) (Rooker et al, 2007).

The development of IEC 61499 was so strongly associated with the HMS research that it made the twin concepts of adaptability and reconfigurability its main focus. However, a full support for dynamic reconfiguration is beyond the scope of the standard (Zoitl et al, 2007).

The major benefit of the IEC 61499 methodology is a separation of concerns (Rooker et al, 2007). The whole application is first programmed as a Function Block (FB) network as in centralized systems. Then, the components of the network are mapped to the devices of the real system where they are executed. This facilitates the movement of functionality from one controller to another (since only the mapping of FBs change, while the original application remains unchanged) and the enhanced support of distribution enables the idea of component-based automation (Rooker et al, 2007).

Support for the IEC 61499 standard is not yet widespread and this is because its development is still relatively recent (Black and Vyatkin, 2009). While hardware support for the standard is still very limited, a number of software platforms for the implementation of the standard have been developed. The first implemented IEC 61499 execution environment is the FBRT (Function Block Run-Time Environment) by James Christensen (Zoitl et al, 2007). This FBRT is an integral part of the Function Block Development Kit (FBDK) also developed by James Christensen of Holobloc Incorpocrated, USA. FBDK is an engineering support tool for the execution of the IEC 61499 standard which is available for free on the Holobloc website. It was used for the world's first factory installation of IEC 61499 by Tait Control Systems Limited in New Zealand (Vyatkin, 2007). Other execution environments for the standard have since been developed and these include O^{3}NEIDA Workbench (now Fbench, promoted by Dr. Valeriy Vyatkin), CORFU ESS (CORFU Engineering Support System developed by Prof Kleanthis Thramboulidis), IsaGRAPH v5.0 (the first commercial software compliant with the standard), etc.
Some research work has been done on the use of the standard. Thramboulidis et al (2004) used real-time UML as a meta-model between the design models of IEC 61499 and their implementation models to support dynamic reconfiguration of

control applications. The TORERO research project focuses on plug and play, and self-reconfiguration of field devices using IEC 61499 (TORERO, [s.a]).
The standard is not, however, without its own critics. Lewis (2001) mentioned the non-adoption of some of the concepts of object-oriented software technology, such as inheritance, as part of the criticism leveled against the standard. Vyatkin (2006) studied the impact of the standard in the industry and concluded that, unless control systems are developed based on this standard to solve the current need of consumers, the concept might as well be doomed to die.

2.5 Conclusion

Different manufacturing systems and the methods available for controlling them have been reviewed in this chapter. Some of the control methods can be applied to more than one type of manufacturing system. The methods of multi-agent systems and distributed control based on the IEC 61499 standard will be used for the reconfigurable control of the feeder. Figure 2.4 illustrates the manufacturing systems and the control methods commonly associated with them.

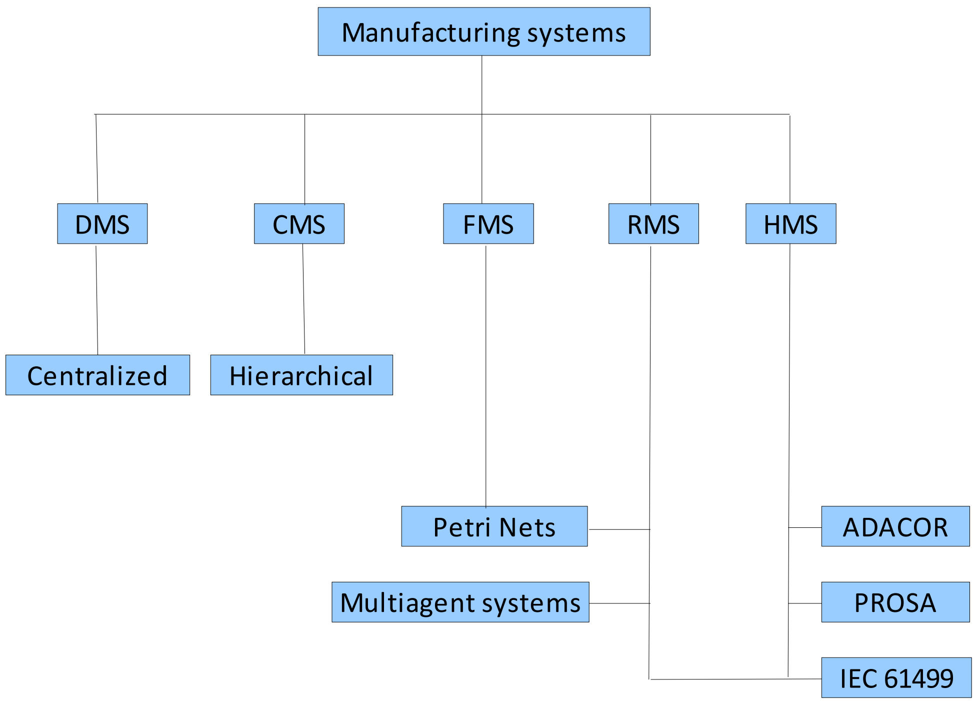


Figure 2.4 Manufacturing systems and their control methods

3 Description of the case study

The assembly system dealt with in this work is a welding assembly system. The adopted configuration is based on the conceptual design developed by Sequira (2008) for a fixture-based reconfigurable spot welding system. In his work, Sequira (2008) proposed four layouts and decided on "the loose pallet assembly" which consists of a central round robin main loop, with modular in-feed and out-feed conveyor units, allowing for the implementation of parallel loops. This is the configuration used in this work with some slight, but cost-saving, modifications.

3.1 Assembly system overview

The assembly system is being constructed at Stellenbosch University. It consists of four main systems and a conveyor. The four systems are the pallet magazine, the feeding system, the welding system (automatic welding machine) and an inspection/removal system. The spatial arrangement of the assembly system is shown in figure 3.1. The feeding system consists of a number of singulation units, a camera and a robot, while the inspection/removal system consists of a camera and a robot. There are two feeding systems in the assembly system.

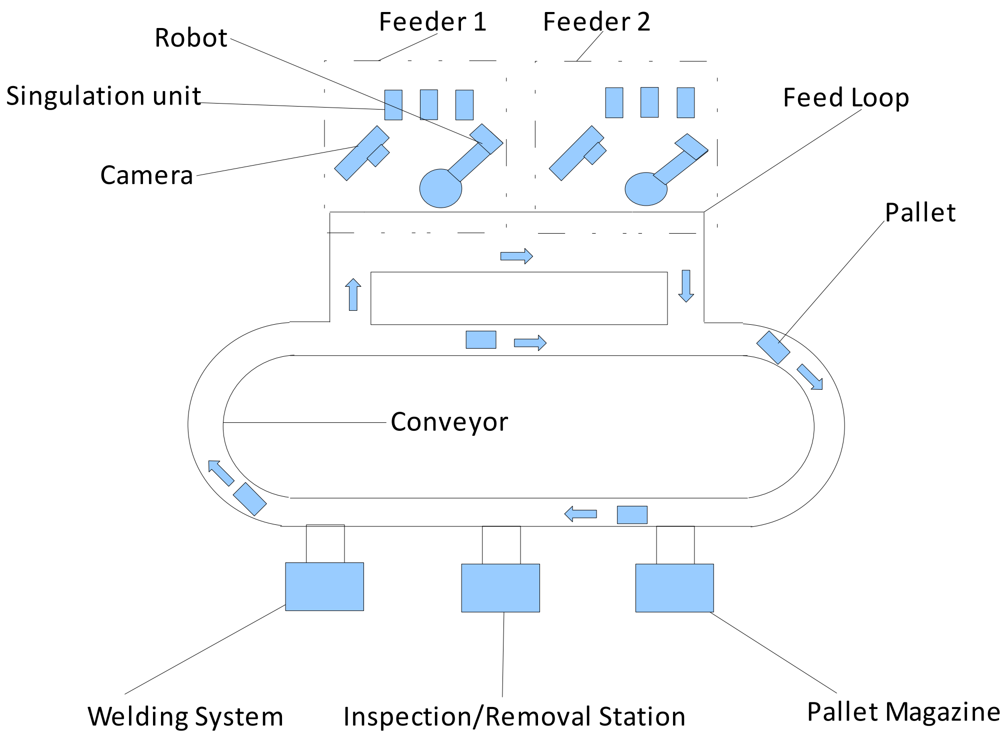


Figure 3.1 Spatial arrangement of the welding assembly system

The assembly system in its current configuration is designed to weld some component parts of CBI's Q-frame circuit breaker. The components of the circuit breaker are shown in figure 3.2. The areas marked with circles represent the points where the components will be welded. The sizes of the components range from 10 mm to 60 mm, which are the lengths of the two pigtails. The dimensions of the moving contact and the arc runner are 27 x 8 x 12 mm and 42.6 x 9.8 x 18.4 mm respectively. The pigtail diameters are between 2.5 mm and 4 mm, while the length of the coil is about 21 mm with an external diameter of 9 mm. The variations amongst the Q-frame breakers are minor and they can all be assembled using the current assembly system configuration. Types other than the Q-frame will require reconfiguration (e.g. changes to grippers and welder electrodes), but the particulars of that reconfiguration have yet to be determined. The type of reconfiguration in the present study is aimed at the ability to replace or add subsystems to adjust production capacity and/or material handling routes.

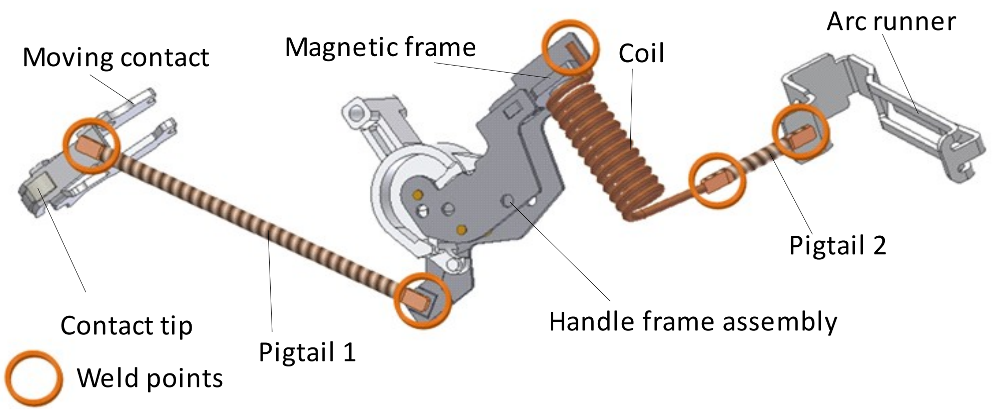


Figure 3.2 Components of circuit breaker

The welding assembly system should typically operate as follows:

The pallet magazine, which is capable of storing or releasing pallets to the system, supplies a pallet based on the type of fixture needed on the pallet. The conveyor transfers the pallet from one station to the other. First, the pallet moves to the feeding stations where parts are loaded onto the fixture. Then the pallet moves to the inspection station to confirm that all parts are present and correctly located. It then moves to the welding station where the parts are welded. Upon completion of welding, the pallet moves to the inspection and removal station where the necessary inspection is done and the assembled part is offloaded. When the production of the current assembly is complete, the pallets used for that assembly are returned to the pallet magazine, which removes them from the conveyor system.

3.2 Pallet magazine

The pallet magazine was designed by Burger (2009) and is shown in figure 3.3. The design, advantages and reasons for the choice of each component during the design process are fully described in Burger (2009). The pallet magazine consists of a rotary magazine assembly, a safety screen assembly, a conveyor assembly, a linear drive and a piston. The rotary magazine assembly consists of three magazines, which can store three different fixture-type pallets. The different fixture-types determine the variants of the final product assembly produced. When a particular type of pallet is required, the magazine assembly is positioned by the magazine assembly motor (mounted below the magazine assembly). This motor is a geared servo motor, which is controlled by an inverter drive. The inverter drive is connected to the main controller via PROFIBUS.

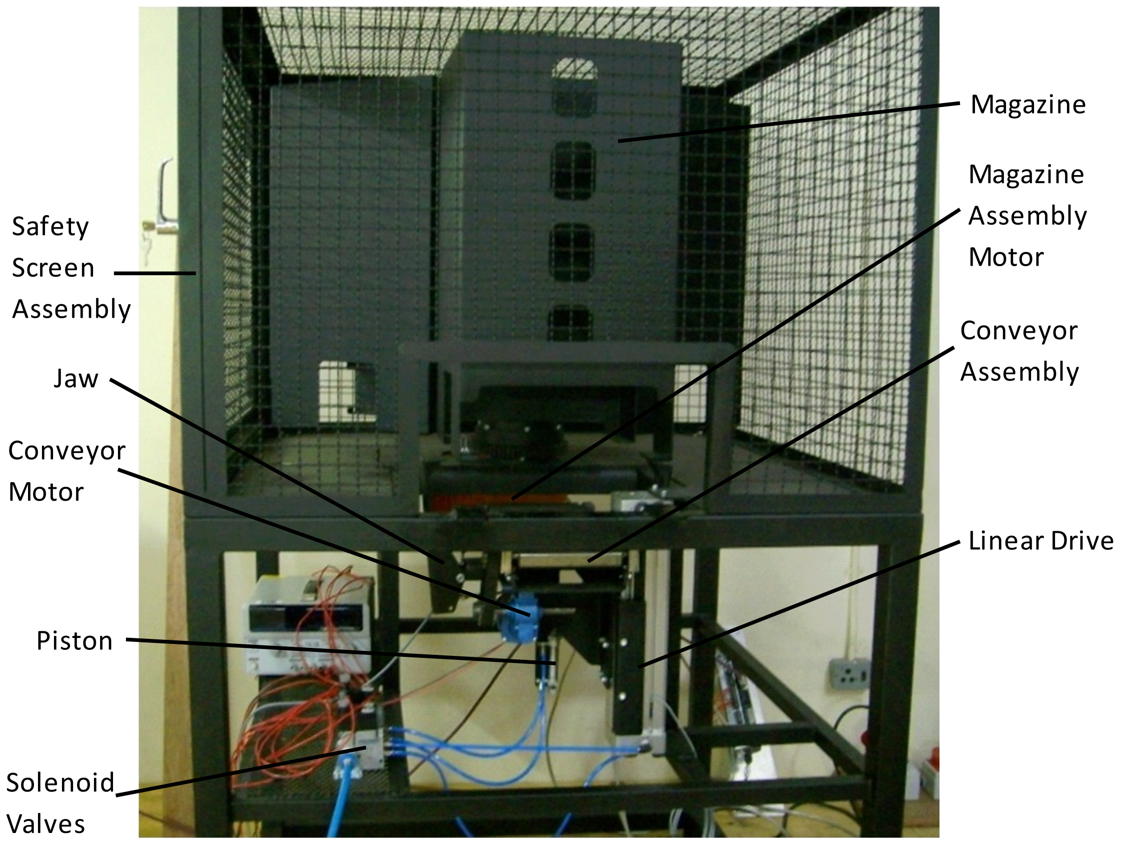


Figure 3.3 Pallet magazine

The conveyor assembly is responsible for moving the pallet to be transferred into or out of the magazine. The direction of rotation of the conveyor is determined by the conveyor motor, which is a DC geared motor capable of turning in the clockwise or counterclockwise direction depending on the direction of power connected to it. The conveyor assembly is connected to the linear drive, which is responsible for raising and dropping the conveyor assembly. The jaws form a part of the conveyor assembly. The purpose of the jaws is to lift the stack of pallets in the magazine when a pallet is to be transferred into the magazine. However, if a pallet is to be released from the magazine, the jaws lift all the pallets except the lowest one.

The jaws are opened and closed by a pneumatic piston. This piston is connected to a solenoid valve, which is a 4/2 way valve. The linear drive, on the other hand, is connected to two 3/2 way valves. These valves are used to lift and drop the linear drive. When the conveyor assembly is to be raised, one of the valves is opened and the other is closed, while if the conveyor assembly is to be dropped, the states of the valves are reversed.

The pallet magazine can perform two functions, namely loading of pallets on to the conveyor system, which shall be referred to as “uploading” pallets, and removing pallets from the conveyor system, which shall be referred to as “offloading” pallets.

3.2.1 Uploading pallets

The procedure for uploading pallets is as follows:

First, the pallet magazine is informed by the higher level controller to upload a pallet. Then, the following sequence of operation occurs, as shown in figure 3.4:

1. Position the magazine – The magazine assembly motor positions one of the magazines above the conveyor assembly. As stated earlier, there is provision for 3 different kinds of pallet depending on the type of fixture on it, with each set of pallets stored in one of the three magazines. Therefore, the right magazine is positioned depending on the type of pallet required.

2. Raise the conveyor assembly – The conveyor assembly is raised to a height midway along the length of the linear drive. At this height, the jaws (when closed) are between the lowest two pallets.

3. Close the jaws – The jaws are closed. At this point, the jaws are between the lowest two pallets in the magazine.

4. Raise the conveyor assembly further – The conveyor assembly is raised further till it reaches the full length of the linear drive. The effect of this is that all the other pallets are lifted, by the jaws, off the lowest pallet

(which is the one that will be uploaded). At this point, the lowest pallet becomes free to move and rests solely on the conveyor.

5. Run the conveyor in the outward direction – Running the conveyor motor clockwise causes the conveyor belt to run in the outward direction. This motion transfers the pallet out of the magazine.

A proximity sensor attached to the magazine indicates when the pallet leaves the magazine and then the pallet magazine returns to the initial state by stopping the conveyor motor, dropping the conveyor assembly and opening the jaws.

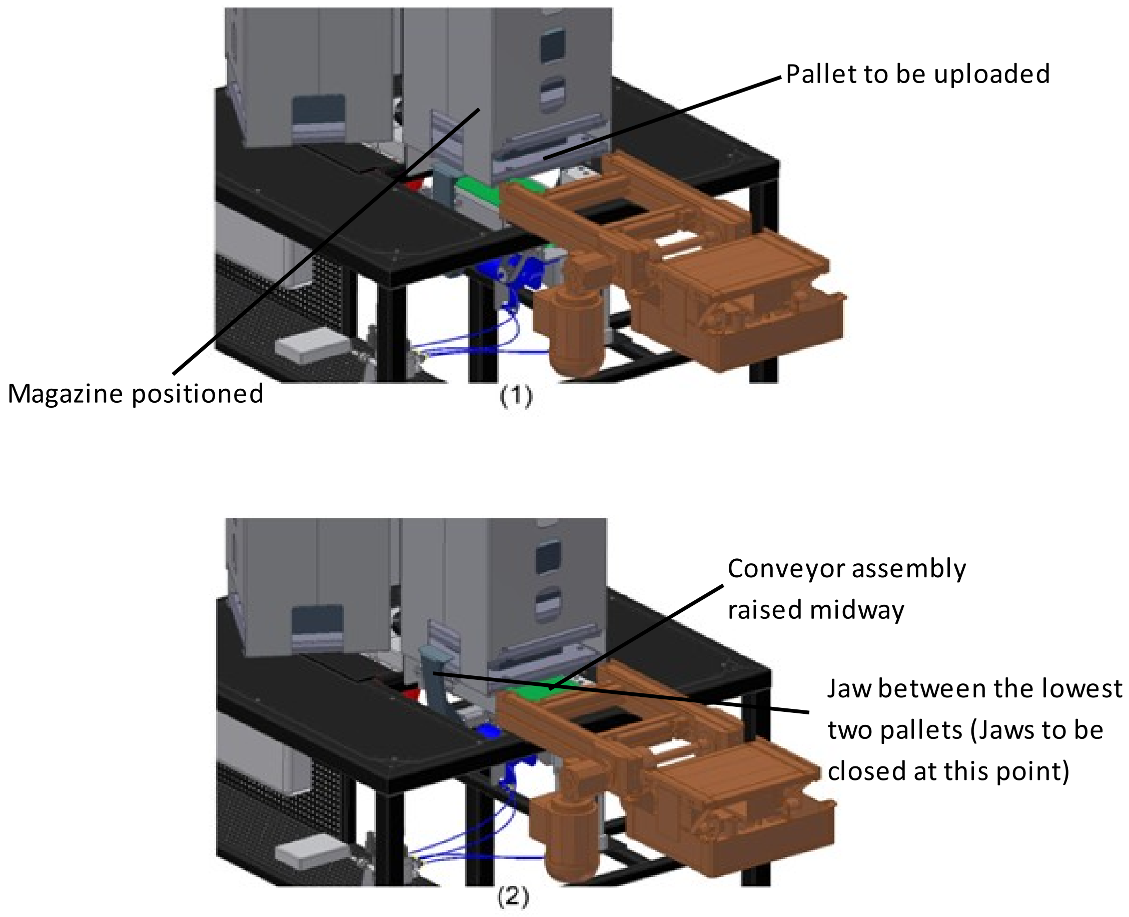
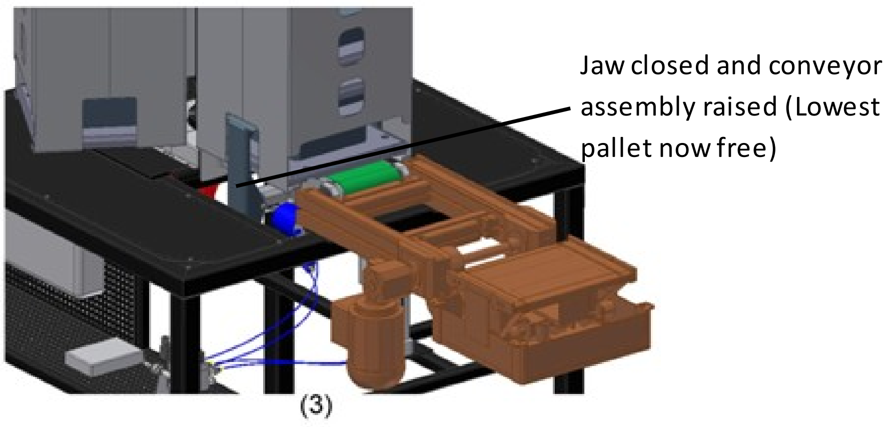
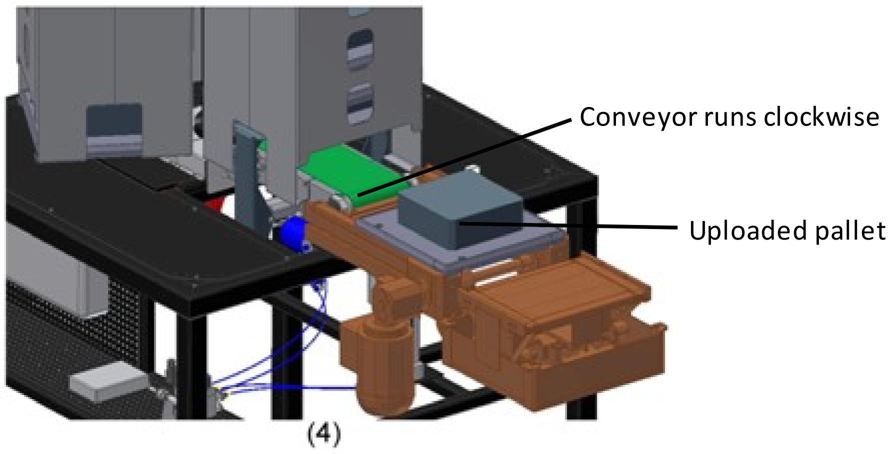

Pallet to be uploaded
Magazine positioned
(1)
Conveyor assembly raised midway
Jaw between the lowest two pallets (Jaws to be closed at this point)
(2)

Figure 3.4a Uploading pallets (Burger, 2009)


Jaw closed and conveyor assembly raised (Lowest pallet now free)
(3)



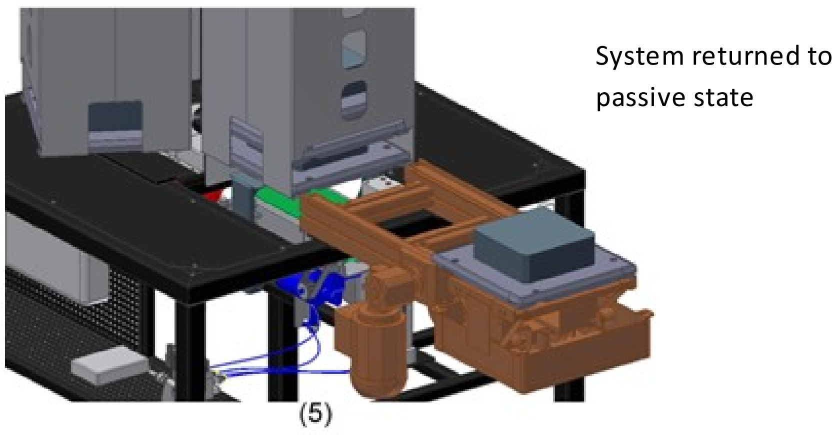


Figure 3.4b Uploading pallets (Burger, 2009)

3.2.2 Offloading pallets

The procedure for offloading pallets is as follows:

First, the pallet magazine is informed by the higher level controller to offload a pallet. The presence of the pallet to be offloaded is detected using a proximity sensor on the conveyor leading to the pallet magazine, and then the following sequence of operation occurs as shown in figure 3.5:

1. Position the magazine – The magazine is positioned based on the type of pallet to be offloaded.
2. Close the jaws – The jaws are closed so that at the next step, when the conveyor is raised, the jaws lift all the pallets in the magazine.
3. Raise the conveyor assembly – The conveyor assembly is raised to the full length of the linear drive so that it is at the same height as the pallet to be offloaded. At the same time, the pallets in the magazine are lifted by the jaws.
4. Run the conveyor in the inward direction – Running the conveyor motor counterclockwise causes the conveyor belt to run in the inward direction. This motion transfers the pallet into the magazine. The proximity sensor attached to the magazine detects the presence of the pallet in the magazine. The conveyor motor stops running when the pallet arrives in the magazine.
5. Drop the conveyor assembly – The conveyor assembly is dropped so that the pallets supported by the jaws are brought to rest on the newly arrived pallet. The jaws can now be opened.

The pallet magazine is then back at its initial state where the conveyor assembly is dropped and the jaws are open.

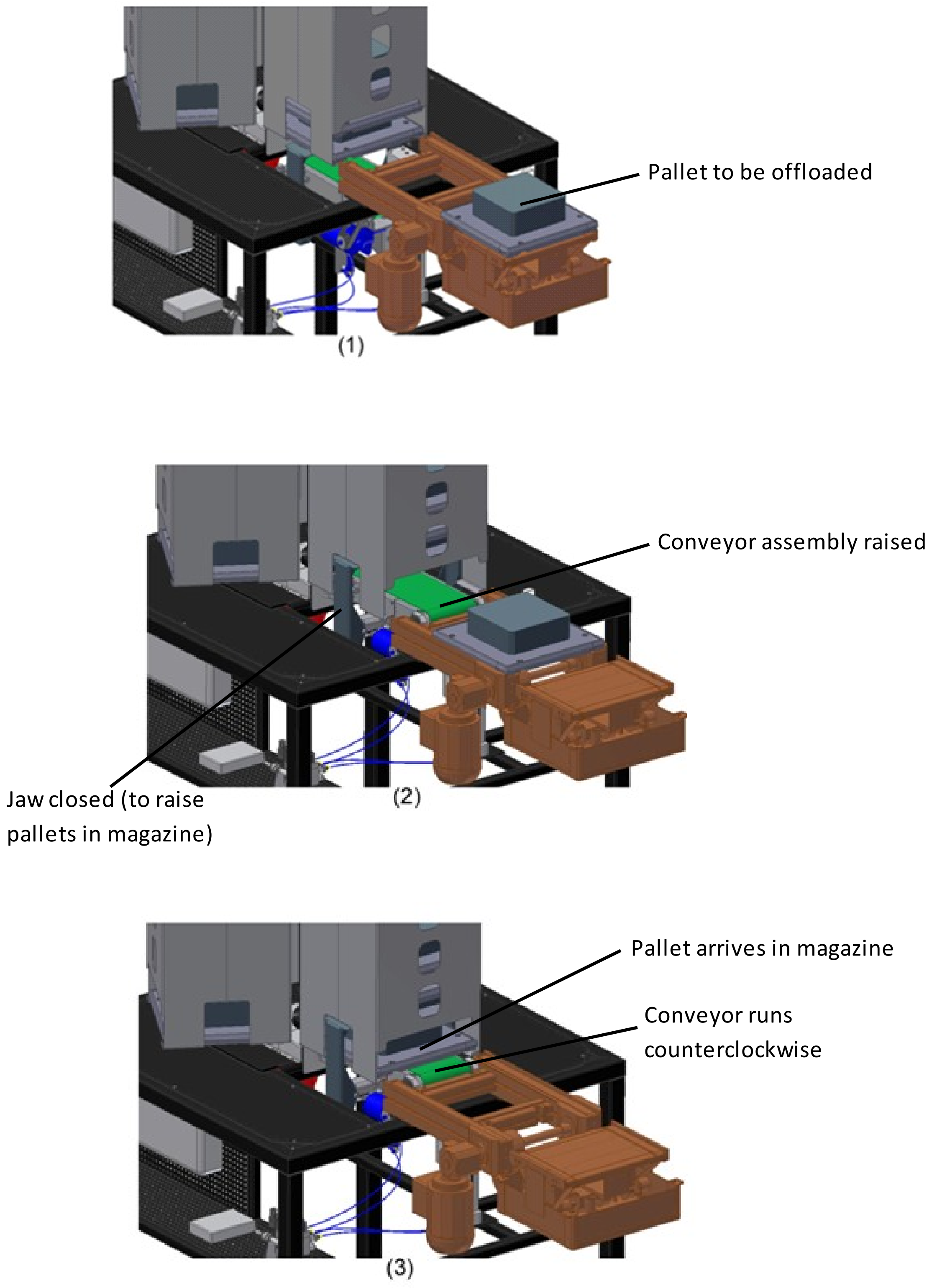
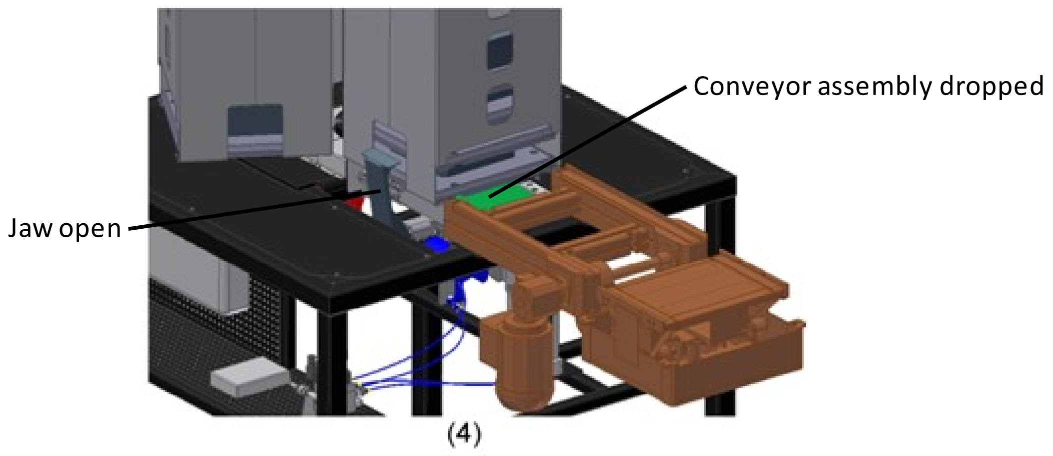

Pallet to be offloaded
(1)
Conveyor assembly raised
Jaw closed (to raise pallets in magazine)
(2)
Pallet arrives in magazine
Conveyor runs counterclockwise
(3)

Figure 3.5a Offloading pallets (Burger, 2009)



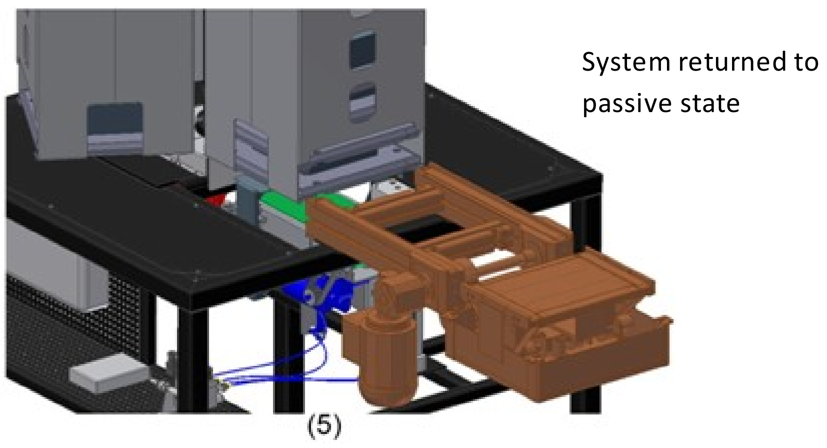


Figure 3.5b Offloading pallets (Burger, 2009)

3.3 Feeding system

The feeding system is an aggregate of three major subsystems, namely the singulation unit, the feeder camera and the feeder robot. These three subsystems work together to perform the feeding system function of loading parts into the fixture on the pallet. The singulation unit transfers parts from a hopper and separates or singulates them so that they are not tangled. The parts are inspected by the camera which returns the position of the parts' pick point. The robot uses this pick point position in picking the part and placing it on the pallet.

3.3.1 Singulation unit

The singulation unit was designed by Strauss (2009) and is shown in figure 3.6. The design, advantages and reasons for the choice of each component during the design process are fully described in Strauss (2009). The singulation unit consists of a hopper, a hopper conveyor, two transfer conveyors running at different speeds and an inspection section. The original design by Strauss (2009) had a flipping section that uses three pneumatic pistons and a stepper motor. The purpose of the flipping section was to orientate the part, but this was removed due to concerns about its reliability. The flipping section was then replaced by air nozzles which reject unwanted parts using air jets.

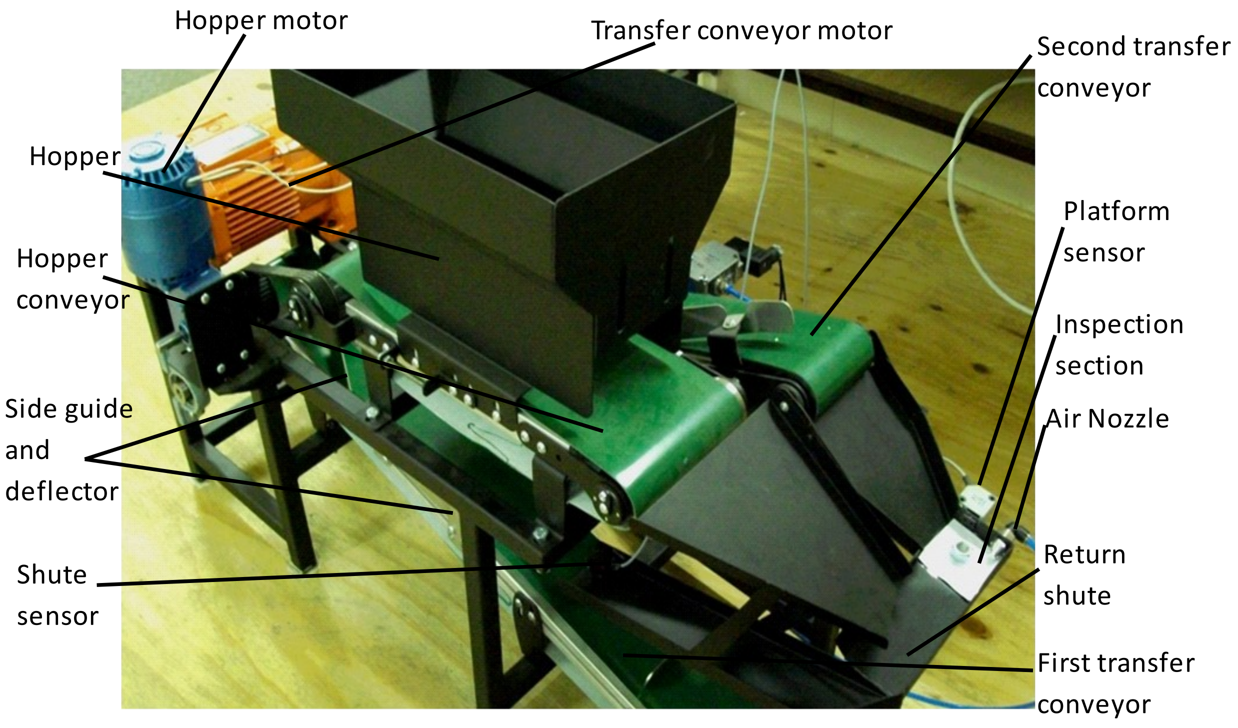


Figure 3.6 Singulation unit

Parts to be loaded are transferred manually from the parts bin to the hopper. The collection of parts in the hopper is unsorted and may, therefore, be tangled and intertwined. As the hopper conveyor, which is driven by the hopper motor, runs, parts drop off the hopper conveyor belt onto the first transfer conveyor through a shute. There is a shute sensor which is used to detect the passage of parts along the shute. This sensor serves two diagnostic purposes: it detects whether parts have actually dropped from the hopper to the transfer conveyor, and it also detects whether any part rejected from the inspection platform has passed through the return shute. Along the length of the first transfer conveyor are side guides and a deflector. These guides prevent the parts from dropping off the side of the conveyor, and at the end of the first transfer conveyor is the deflector which directs the parts onto the second transfer conveyor. There is a “transfer sensor” (not shown in the figure) at this transfer point from the first transfer conveyor to the second. This sensor tracks the parts that have moved from the first to the second transfer conveyor. The first and second transfer conveyors are driven by a three-phase AC motor, but are geared to run at different speeds with the second conveyor running faster than the first.

The reason for this speed difference between the transfer conveyors is to create gaps between the parts. Along the length of the second transfer conveyor are deflectors which will only allow single parts to pass through. There is also a wiper blade which allows only parts of a particular maximum height to pass through. Any parts which are still tangled are pushed back onto the first transfer conveyor by the deflectors and the wiper blade. There is also another sensor here (called “circulating sensor”, and is not shown in the figure) which detects the parts returning (or circulating) from the second to the first transfer conveyor. When the part reaches the end of the second transfer conveyor, it slides on to the inspection station. An optical sensor detects the arrival of the part and triggers the stoppage of the motor driving the transfer conveyors.

Part inspection is done by the camera and the part is only rejected if the camera determines that the part is not the one required to be loaded or if the part is in an unacceptable pose. The singulation unit rejects parts by ejecting a high pressure air jet from an air nozzle which drives the part back, along the return shute, to the first transfer conveyor.

3.3.2 Feeder camera

The feeder camera is a Cognex DVT Legend 540 vision sensor shown in figure 3.7. The vision sensor has an image resolution of 640 x 480 pixels and a grayscale CCD (charge-coupled device) for image detection. It has exposure times of between 10µs to 1s and an isolated power supply of 24V DC. The camera has two RJ-45 ports, one for power supply and the other for communication over Ethernet. There is also a separate digital I/O with 8 configurable inputs and outputs that can be used to send commands to the vision system.

The feeder camera inspects the part in the inspection section of the singulation unit. If the camera confirms that the inspected part is the part required to be loaded, it then determines the position of the pick point of the part. This information is what the robot uses to pick the part. On the other hand, if the part is not the part required or the part not in a collectible orientation, it informs the singulation unit to reject the part.

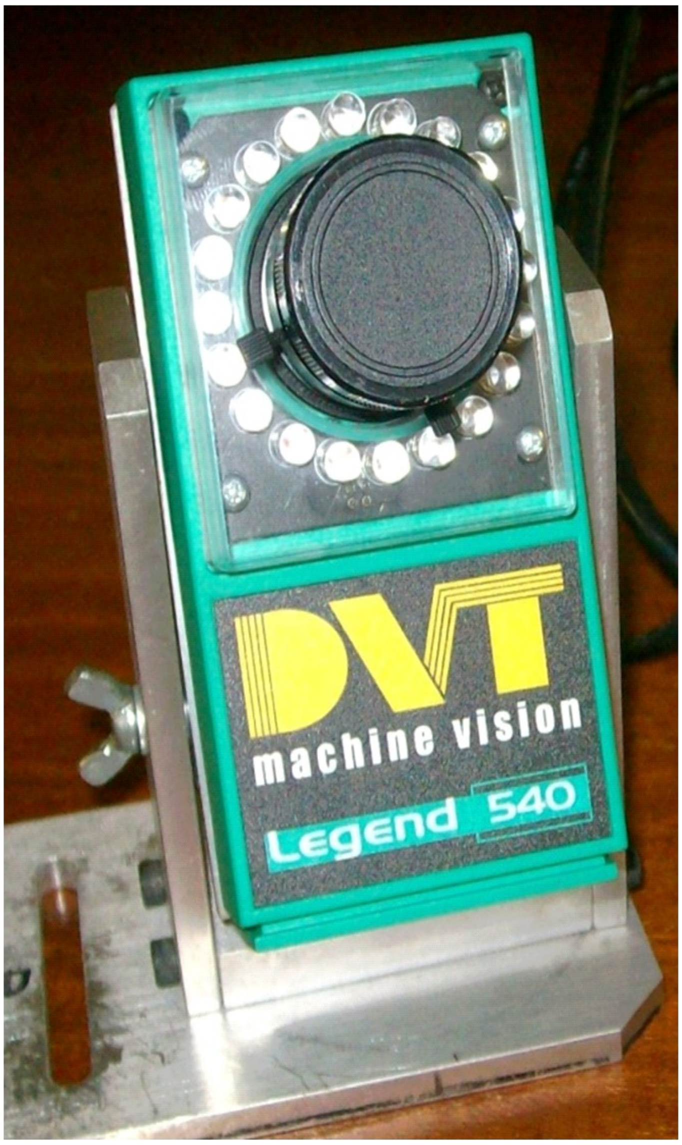


Figure 3.7 Cognex DVT vision sensor

3.3.3 Feeder robot

The feeder robot is a six-degree of freedom RTX robot manufactured by Universal Machine Intelligence. Figure 3.8 shows the robot as depicted in its manual, "Introducing RTX" (UMI, 1986). The robot has an upper arm, a lower arm at which end is a gripper and a vertical linear slideway. It also has three joints: the shoulder, elbow and wrist joints. The upper arm is driven by the shoulder motor, the lower arm is driven by the elbow motor, while the wrist is controlled by two motors which enable it to perform both pitch and roll motions.

The role of the robot in the feeding system is to pick parts based on the position given by the camera and place them on to the pallet. The coordinate space of the robot is measured in millimeters, which is the same unit used for the transformation coordinate for the camera. The camera only supplies dimensions in two coordinates while the z-coordinate of the camera is fixed.

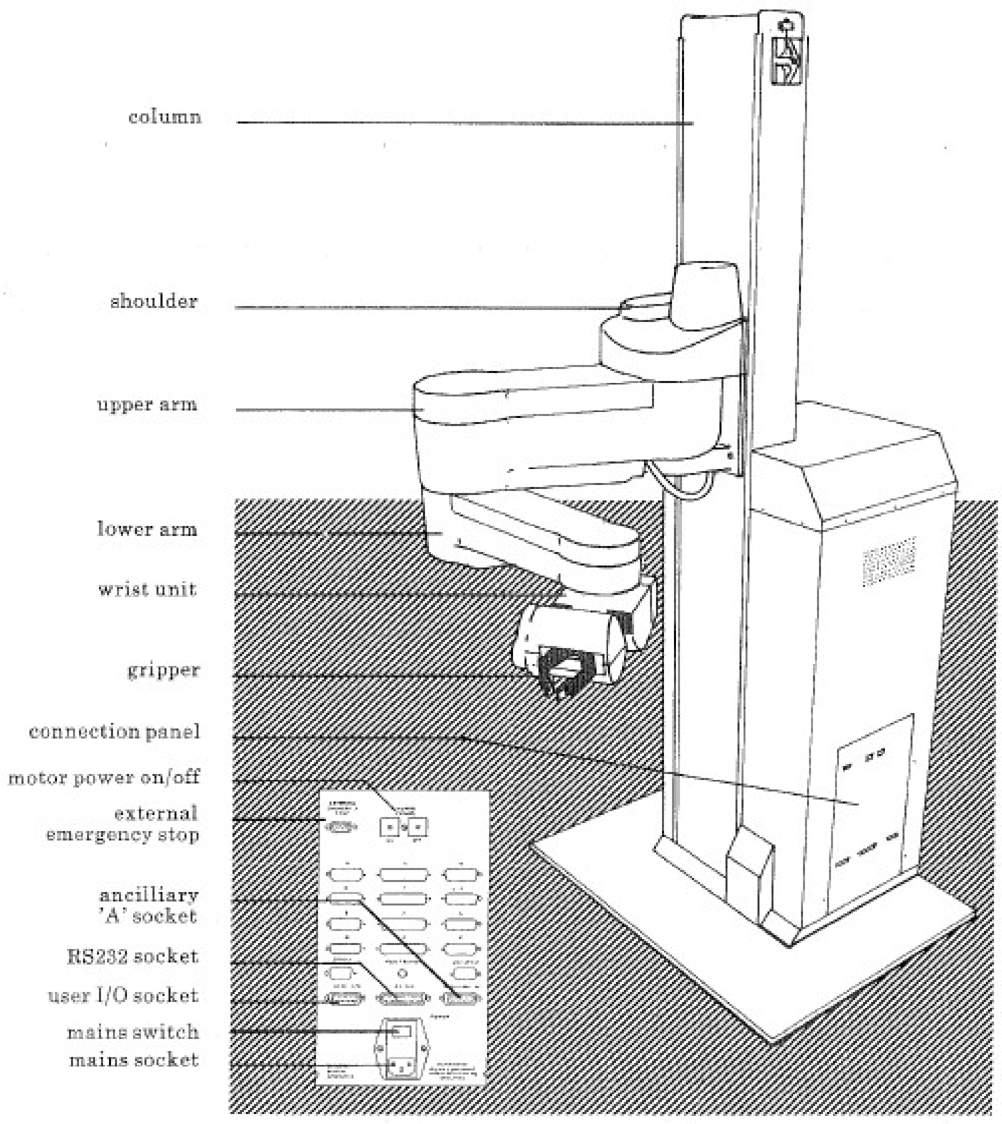


Figure 3.8 RTX robot (UMI, 1986)

3.4 Other subsystems

The other subsystems of the welding assembly system are: the inspection and removal system, the welding system and the conveyor. The inspection and removal system is almost the same as the feeding system except for the absence of a singulation unit. The inspection and removal station is where inspection is done on parts loaded on the pallet fixture before welding, and also where the final inspection on the welded parts is carried out before removal. Here, the welded parts are inspected for defects and they are transferred to the assembly bin if the inspection is successful, or to the rejection bin if the inspection is not successful.

The welding system spot-welds the parts loaded on to the pallet. However, due to the high cost of an automated welding system, no physical welding system was purchased. Its actions of welding in this study case will only be simulated. The conveyor to be used is a Rexroth TS2 Plus transfer system. It has several motors, an ID40 RFID system, a number of sensors connected on an AS-i network and a PROFIBUS communication interface. The incomplete conveyor is shown in figure 3.9.

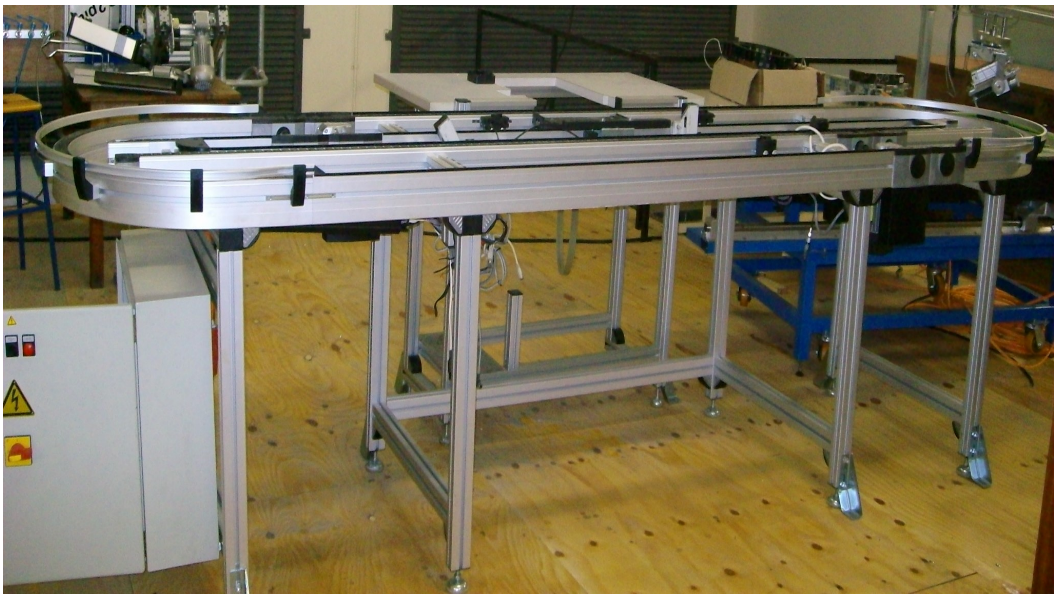


Figure 3.9 Rexroth TS2 Plus conveyor

4 Low level control of the subsystems

The control of the subsystems in this work is divided into two: the low level control and the high level control. The low level control directly interacts with the hardware of the subsystems by reading from sensors, and activating the actuators of the subsystems. This control is covered in this chapter. The high level control, which is considered for the feeding system, is termed reconfigurable control and is covered in chapter 5. The low level control of the subsystems is to be implemented using a PC and a PLC. The subsystems to be controlled are the pallet magazine and the feeding system components which include the singulation unit, the camera and the robot.

4.1 Pallet magazine

The hardware components of the pallet magazine were described in section 3.2. The actuators responsible for the movement of these components include:

- the magazine assembly motor which is used to position the magazine assembly,
- the conveyor motor which runs the conveyor,
- the solenoid valves which regulate the flow of air into the linear drive and the piston which operates the jaws.

We first discuss the control of the components common to the PLC and PC controllers, and then discuss the control as done using the PLC and the PC, respectively.

4.1.1 Magazine assembly motor

The magazine assembly motor is a geared servo motor from SEW EURODRIVE (KAF37 DS56M). In order to control the motor, a drive inverter, the SEW MOVIDRIVE MDX61B0005-5A3-4-0T, was used. The characteristics of the motor utilized in order to position the motor, as given by the manufacturer, are:

- gear ratio of motor, which is 29.96:1
- the gear teeth values, which the manufacturer gives as pinion 23T, gear 31T, second gear pinion shaft 18T, gearwheel 97T, rose 8T and bevel 33T.

In the control of the pallet magazine, the communication between the controller and the Movidrive inverter is over PROFIBUS, which is a commonly used communication standard in the industry. The drive inverter is able to connect to a PROFIBUS network using a DFP21B PROFIBUS module. The PROFIBUS address of the drive inverter was set to number 4. The magazine assembly motor has an internal resolver which serves as its encoder. With this resolver, the drive inverter is able to monitor and report the actual position of the motor. Also, because of the direct connection between the motor and the magazine assembly (which is termed as “positive connection”), there is no need for an external encoder. For these reasons, a resolver module, the DER11B, was installed on the drive inverter instead of an encoder module.

Setting up the drive inverter involved the use of IPOS which is a programmable positioning language developed by the drive manufacturer. The Modulo positioning tool was specifically used in this case because what is to be done here is rotational positioning (not linear positioning). The Modulo positioning tool uses a reference point, defined by a reference cam, and a reference travel type to specify the initial position of the motor. In order to use the Modulo positioning tool, the drive inverter is first set up using a keypad. The data of the magazine assembly motor are entered into the inverter. These data values include the voltage rating, the power, the maximum current, etc, of the motor. The next step is to configure the inverter from a computer using the Movitools software which is also supplied by the manufacturer. A USB/Serial converter is used to connect the drive inverter to the computer for automatic start up. Additional data, which include ramp up and ramp down times, maximum moment of inertia, etc, are supplied.

The final step is to set up the Modulo positioning tool. To do this, the drive inverter is first wired based on the wiring diagram shown in appendix A.1. The Modulo positioning tool automatically configures the first four digital inputs as follows: DI00 is Control Inhibit, DI01 is Enable, DI02 is Reset and DI03 is Reference Cam Input. In this work, there is no reference cam, so DI03 was not wired. Instead, a reference travel type 5 was chosen. With a reference travel type 5, there is no reference travel and the current position of the motor is taken as the reference point. The values of the gear teeth and the gear ratio were used to calculate the fractional ratio of the gear box. These values were entered as 33077 for “numerator” and 1104 for “denominator” as required in the Modulo positioning tool. The derivation of this ratio of 33077:1104, which translates to the actual gear box ratio (i.e. 29.96:1), is given in appendix A.2.

From the IPOS program of the Modulo positioning tool, there are different ways by which the positioning can be done, but the “automatic positioning with optimization” method was chosen. This method is an intelligent positioning option in which the motor takes the shortest route from its present position to the target position. An illustrative example is the case where the motor is required to move to a 315-degree position when its current position is 20 degrees. With the “automatic positioning with optimization” option, the motor moves in the counterclockwise direction instead of the clockwise direction which is the default. This is because the counterclockwise direction is the shorter route to the target position. However, the disadvantage of the method is that the gear ratio limits the maximum position the motor can attain. Although it is not likely in our case, if the motor exceeds the maximum angle of rotation (about 1176 degrees approx.), then the rotation will have to occur along the longer route.

With the configuration of the Modulo positioning tool, only six process data (PD) words are required by the inverter for positioning the motor. The six output data words represent the Control Word 2, the Target Position (high word), the Target Position (low word), the Setpoint Speed, the Acceleration Ramp and the Deceleration Ramp respectively. On the other hand, the six input data words, which will be returned by the inverter, represent the Status Word, the Actual Position (high word), the Actual Position (low word), the Actual Speed, the Active current and the Device Utilization.

The target position was set, in the Modulo positioning tool, to be measured in degrees. Other units of position measurement allowed are tenth of degree and encoder count. The actual position of the motor returned by the inverter is in encoder count. The method used in this work to determine whether the motor positioning is complete or not is to check the fourth bit of the status word (i.e. process input data 1). This bit represents Target Position Reached, and it has a value of 1 when the target position is reached.

4.1.2 Conveyor motor

The conveyor motor, which drives the conveyor in the pallet magazine, is a 12V DC motor. There are three states in the control of this motor: it runs in the outward direction when a pallet in the magazine is to be uploaded, it runs in the inward direction when a pallet is to be offloaded and it is stationary when the pallet has arrived or left the magazine. Two relays are used in order to control the conveyor motor. The relays are connected to the motor as shown in figure 4.1. The first relay labeled R4 is called the enabling relay while the one labeled R5 is called the direction relay for reasons explained below. The motor is connected to the COM (common) of relay R4, while the N.O. (Normally Open) of relay R4 is connected to the COM of relay R5. Relay R5 is connected to the voltage supply line in the following order: the N.O. of R5 is connected “normally” so that the positive terminal of the motor connects to the positive end of the 12V power supply, while the negative terminal of the motor connects to 0V or the negative end of the 12V power supply. The N.C. (Normally Closed) of R5 is then connected in the reverse direction so that the positive terminal of the motor connects to the negative end of the power supply and the negative terminal of the motor connects to the positive end of the 12V power supply.

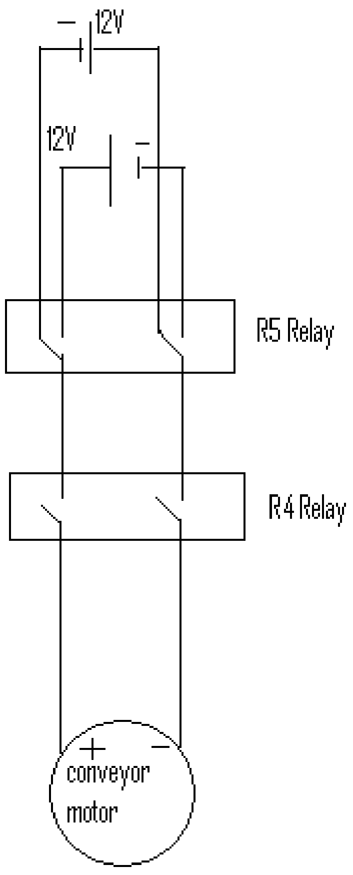


Figure 4.1 Conveyor motor connection

Based on this connection, when the relay R4 is energized, the conveyor motor is connected to power so it moves either in the forward or reverse direction depending on whether relay R5 is energized or not. However, if relay R4 is not energized, the motor stays stationary notwithstanding whether relay R5 is energized or not. This is because there is no power supply connected to the motor through the N.C. of R4. The dependence of the operation of the motor on the energized state of relay R4 is the reason why R4 is called the enabling relay. In the case of relay R5, when the relay is energized, the motor connects to the N.O. of R5 and so rotates in the forward (or outward) direction, while if the coil is not energized, the motor connects to the N.C. of R5 and therefore rotates in the reverse (or inward) direction. Because relay R5 determines the direction of rotation of the motor, R5 is called the direction relay.

The motor is controlled using two digital output bits, which are used to energize the two relays respectively. Therefore, in the control of the conveyor motor, when the motor is to run in the outward direction (i.e. forward), the first bit is set to a value of 1 and the second bit is set to 0. When the conveyor motor is to run in the inward direction (i.e. reverse), both bits are set to a value of 1. If the motor is to be stopped, the first bit is set to a value of 0 and there is no need to set the second bit. The use of two relays in controlling the conveyor motor is inherently safe from the risk of a short circuit. This is because at no time will both current carrying lines from the supply voltage reach the motor at the same time, which ordinarily may lead to a short circuit.

4.1.3 Solenoid valves

There are three solenoid valves used in the pallet magazine. One of them is a 4/2 way valve, while the other two are 3/2 way valves. As explained in section 3.2, the 4/2 way valve controls air supply to the pneumatic piston which opens and closes the jaws in the pallet magazine. On the other hand, the two 3/2 way valves control air supply to the linear drive which raises or drops the conveyor assembly. All three valves are connected to relays which can open or close the valves depending on whether the relays are energized or not. The valves are connected to the N.C. (normally closed) terminals of the relays since the supply ports of the valves are open in the initial state, which means that they allow air to flow through them thereby raising the conveyor assembly in the case of the linear drive, and closing the jaws in the case of the pneumatic piston. This is contrary to what is desired in the control where in the initial state, the conveyor assembly should be down while the jaws should be open. This was achieved by connecting the valves to the relays as stated.

The jaws are controlled using one digital output bit which is able to open or close the jaw depending on whether the bit value is zero or one. The linear drive, shown in figure 4.2, is controlled using two digital output bits. To move the slide on the linear drive upwards, the first bit is set to 1 and the second bit is set to 0, while reversing the bits causes the slide to move downwards. There are three proximity sensors along the length of the linear drive which are labeled as top sensor, middle sensor and bottom sensor in figure 4.2. The top, middle and bottom sensors are used to detect when the slide of the linear drive reaches the top, middle and the bottom of the linear drive respectively.

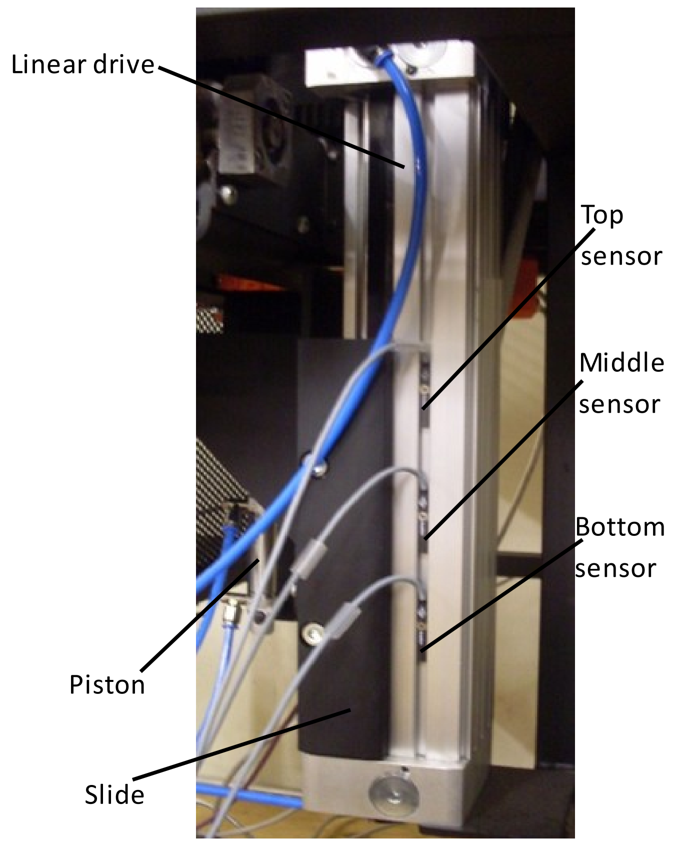


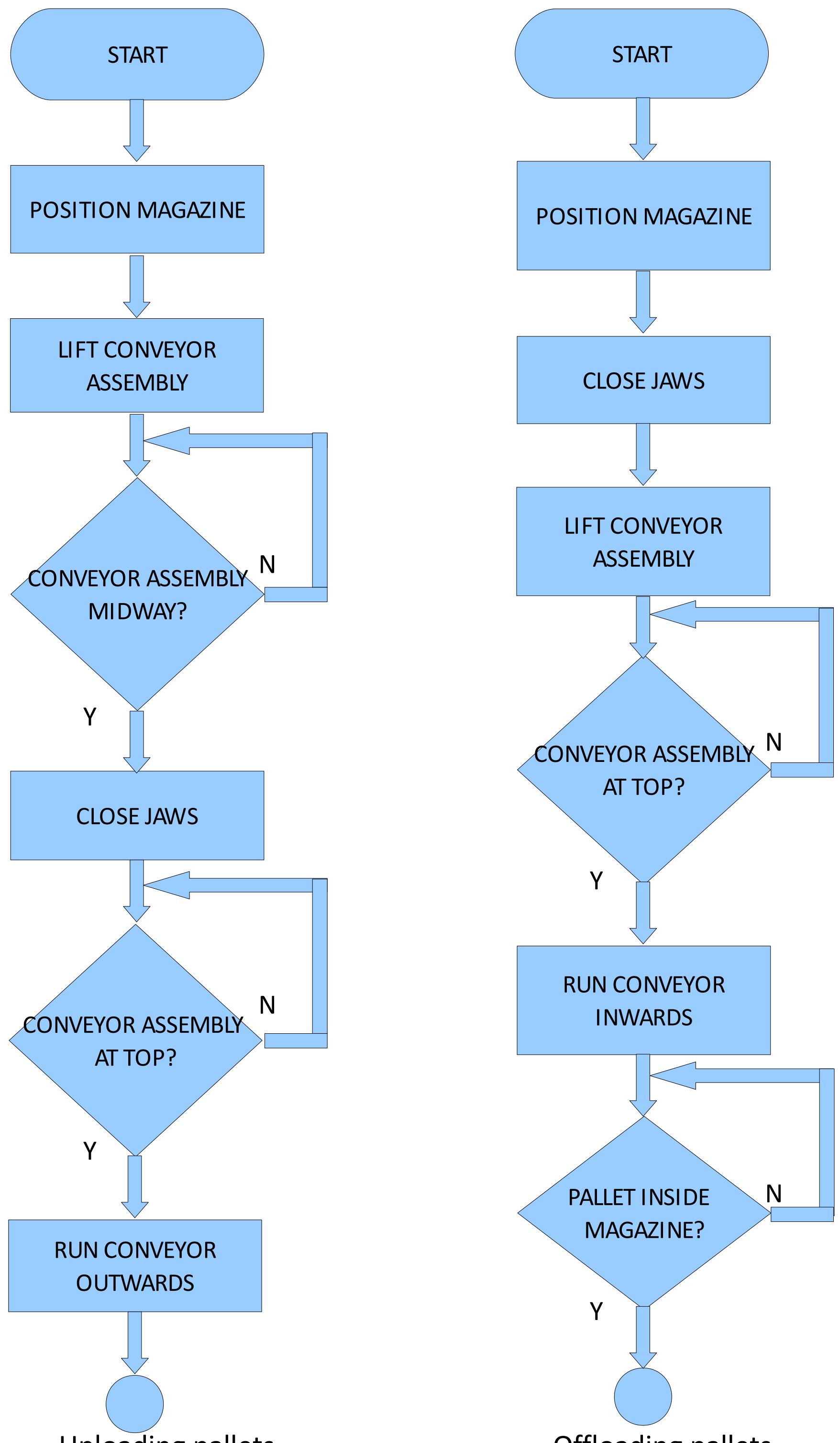
Figure 4.2 Linear drive

4.1.4 PC control of the pallet magazine

The PC control program of the pallet magazine was written using Microsoft Visual C#. Two hardware modules were used with the PC. The first is the Applicom PCIE1500PFB card, which allows the PC to connect to the PROFIBUS communication network, and the second is the Eagle uDAQ Lite data acquisition device. The Applicom card is installed in the PCI Express slot of the PC board which provides a much higher speed of communication than the ordinary PCI slot. The Applicom card enables the PC to act either as a class 1 or a class 2 master on the PROFIBUS network. As a class 1 master, the PC is able to communicate with its configured slaves, while as a class 2 master, the PC only plays a “supervisory” role (i.e. it is used to commission slaves and to obtain diagnostic data from the slaves). In the case of the pallet magazine, the PC is the only master in the PROFIBUS network and acts as a class 1 master. The slave on the PROFIBUS network is the drive inverter which is used to control the magazine assembly motor. The setup of the Applicom card is done using the Applicom software provided by the manufacturer. The GSD file of the slave (i.e. the drive inverter) is installed in the configuration of the Applicom card and the communication was configured for six process data words, as required for the use of the Modulo positioning tool of the drive inverter.

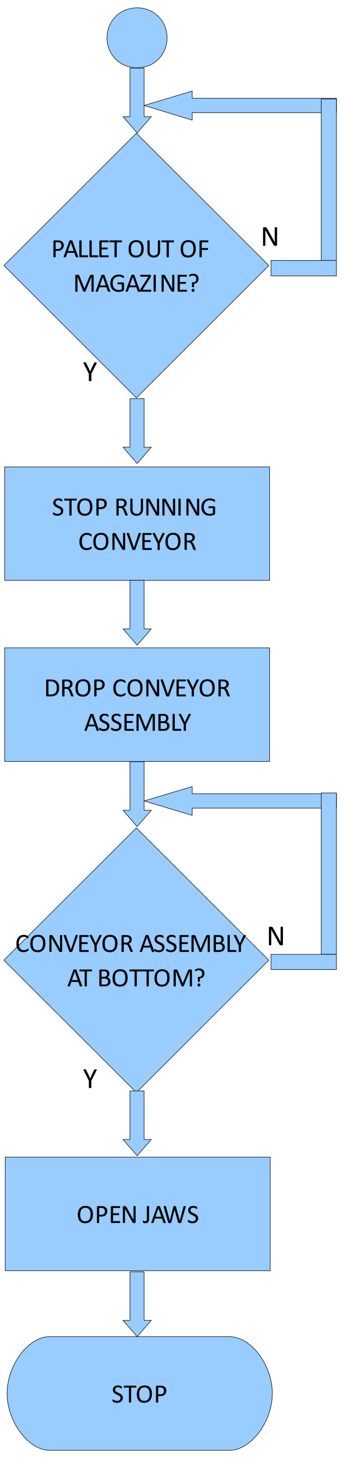
The Eagle uDAQ Lite device, on the other hand, has a USB interface through which it is connected to the PC. It is configured using the Eagle data acquisition software supplied by the manufacturer. The Eagle uDAQ has digital and analog ports but the analog ports were not used in this work. The digital input and output ports are eight bits wide. The Applicom card and the Eagle uDAQ were accessed in the control program through their dll files which were provided by their manufacturers. These dll files were included in the control program as references. The Applicom card dll file gives access to immediate and deferred read and write functions on the PROFIBUS network, and the Eagle uDAQ dll file provides access to read from and write to the digital ports.

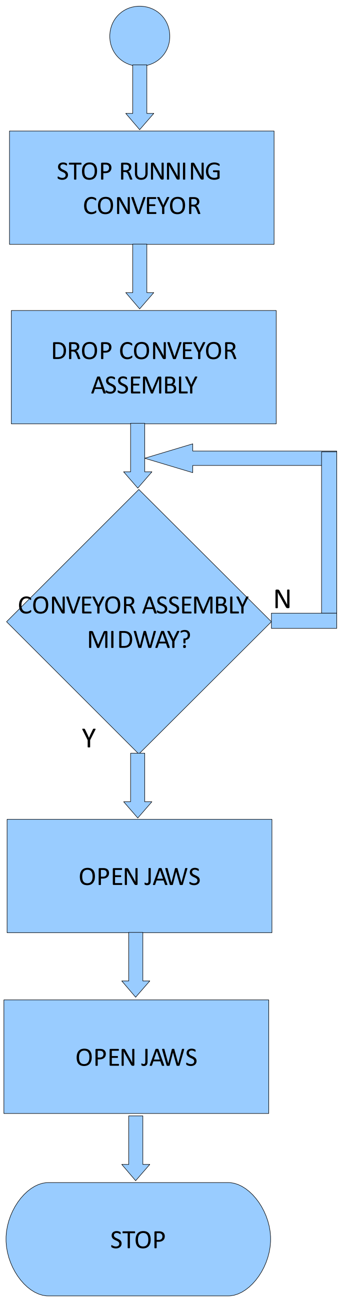
The flowcharts in figure 4.3 show the algorithms for the uploading and offloading of pallets in the pallet magazine. Each of the processes shown in the flowcharts represents a method in the PC control program. For example, position magazine, lift conveyor assembly and open jaws are all methods in the control program. The decision-making methods, which are represented by the diamond shape, involve reading the bit values of the corresponding sensors. For example, whether the conveyor assembly has arrived at the top of the linear drive or not is determined by reading the bit value of the top sensor from the Eagle uDAQ digital input. Reading the digital input returns a corresponding byte value for the 8-bit wide input port, and the value of the bit needed is obtained by carrying out a bit-masking operation. Activating the actuators, for example, to lift conveyor assembly, is also done via the Eagle uDAQ by masking the bit to be changed and writing to the card's digital output. The low output voltage of the Eagle uDAQ device made it necessary to use the Eagle PC-38G, which is a relay adaptor module. The PC-38G has 8 electro-mechanical relay channels, and uses an external 12V DC power supply to energize the relays. Uploading and offloading of pallets in the control program is therefore achieved by calling the individual methods as shown in the flowcharts.



Uploading pallets   Offloading pallets

Figure 4.3a Flowcharts for process flow of pallet magazine





Uploading pallets Offloading pallets

Figure 4.3b Flowcharts for process flow of pallet magazine

4.1.5 PLC control of the pallet magazine

The PLC used for the control of the pallet magazine is a Siemens S300 PLC. The PLC has an integrated PROFIBUS interface. The Siemens STEP 7 program was used to set the hardware configuration of the PLC as well as write the control program using ladder logic. The GSD file for the drive inverter was also installed in the hardware configuration of the PLC just as in the case of the PC. The control program algorithms are also according to the flowcharts in figure 4.3, but the processes in the figure are represented as steps.

The PLC ladder logic program consists of the three function blocks: the upload function block, the offload function block and the position magazine function block (which is used by both the upload and offload function blocks). The position magazine function block uses the STEP7 SFC 14 and SFC 15 special functions to read from and write to the drive inverter over PROFIBUS. As earlier explained, the exchange over PROFIBUS between the master controller and the drive inverter (i.e. the slave) is done using six process data words. For this reason, a UDT (user-defined data type) variable comprising six words was defined and used. The other function blocks, i.e. the upload and the offload function blocks, made use of the position magazine function block to position the magazine assembly motor. The PLC typically executes cyclically so, in order to implement the processes in figure 4.3 as steps, a temporary variable initialized to zero was defined in the function blocks. All the steps in the flowcharts have been labeled consecutively with the first step labeled as zero. Each step only executes if the value of the temporary variable is equal to the number label of that step. Upon completion of each step, the value of the temporary variable is increased by one. By so doing, the PLC executes the program in a stepwise manner.

4.2 Singulation unit

The singulation unit is one of the subsystems within the feeding system. The hardware components of the singulation unit, shown in figure 3.6, were described in section 3.3.1. The actuators within the singulation unit are the hopper motor, the transfer conveyor motor and the solenoid valve for the air nozzle.

4.2.1 Hopper motor

The hopper motor is a PARVALUX SD8 AC motor, which is used to run the hopper conveyor belt. The motor is connected via a 2.5µF capacitor to the power supply. The motor is controlled using only a contactor since there is no need for speed control in our application. The motor is run or stopped by switching this contactor.

4.2.2 Transfer conveyor motor

The transfer conveyor motor is a geared AC motor from SEW EURODRIVE (DR63M4). In order to control the motor, a drive inverter, the SEW MOVITRAC MC07B0005-2B1-4-00, was used. The drive inverter is used only to control the speed of the motor. The drive inverter setup was done using a keypad. The data that had to be provided include the name and type of motor to be controlled, the setpoint speeds to be used, the control operating mode, and various motor data such as voltage, power, frequency, etc. The drive inverter has terminals for binary input and output. The functions of these terminals have already been preset by the manufacturer. The drive inverter can be used to control the motor by writing to the binary inputs.

In the case of the singulation unit, only three of the binary input terminals were used. These are the terminals marked as DI01, DI03 and DI04, which (when they are set) represent the CW (clockwise) rotation, the Enable and the Speed setpoint bits respectively. The transfer conveyors, like the hopper conveyor, only run in one direction in the singulation unit, therefore the DI02 terminal which represents the CCW (counterclockwise) rotation bit was not used. The binary output terminals were used to obtain the status of the drive inverter and the two terminals used are the DO02 and the DO03 terminals, which represent the Brake Released and the Motor Ready bits respectively.

4.2.3 Solenoid valve

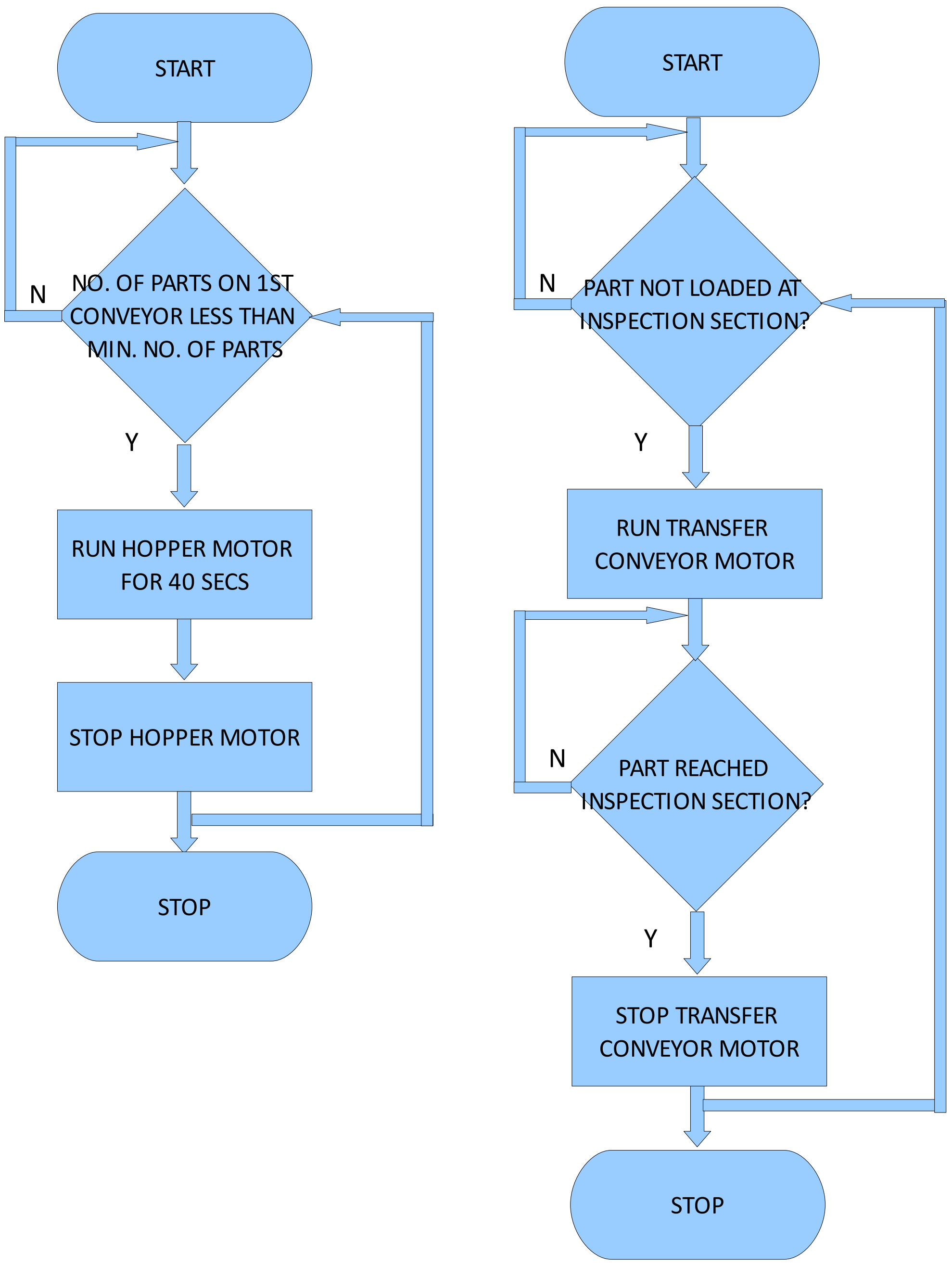
The solenoid valve is used to regulate the flow of air to the air nozzle. The solenoid valve is a 3/2 way valve. The valve is connected to a relay which opens or closes the valve depending on whether the relay is energized or not. This valve is also connected to the N.O. (normally open) of the relay and is controlled using only one digital output bit.

4.2.4 PC control of the singulation unit

The PC control program for the singulation unit was also written using Microsoft Visual C#, as in the case of the pallet magazine. The Eagle uDAQ Lite data acquisition device was used to read from the sensors and write to the actuators. The singulation unit loads parts as well as rejects parts that fail the inspection by the camera. The flowcharts in figure 4.4 show the algorithms for the loading and the rejection of parts. Just as in the case of the pallet magazine, each step is represented in the control program as a method. The decision-making methods involve reading the bit values of the corresponding sensors, while activating the actuators is done via the Eagle uDAQ device by masking the bit to be changed and writing to the card's digital output. The PC-38G relay module was also used. Four sensors, described in section 3.3.1, were used to aid decision-making: the shute sensor, the transfer sensor, the platform sensor and the circulating sensor. The shute sensor is used to detect whether parts are actually supplied from the hopper when the hopper motor is run, and whether parts rejected have passed through the return shute. If no part is detected, it triggers a fault output. The transfer sensor is used to detect parts that move from the first transfer conveyor to the second transfer conveyor, while the platform sensor is used to detect whether a part has arrived at the inspection platform. The circulating sensor is used to detect parts that are circulating back to the first transfer conveyor from the second transfer conveyor. These sensors are used to update the values of three variables used in the control: the number of parts on the first transfer conveyor (x1), the number of parts on the second transfer conveyor (x2), and the number of circulating parts (x3). Every part detected by the shute sensor increases x1 by 1, and every part detected by the transfer sensor decreases x1 by 1 and increases x2 by 1. Every part detected by the platform sensor decreases x2 by 1 and every part that is detected by the circulating sensor increases both x1 and x3 by 1, and decreases x2 by 1. Variable x3 is a diagnostic variable and it triggers a warning output if its value is greater than a minimum (5, in our case), and its value is reset to zero after every 10 parts loaded.

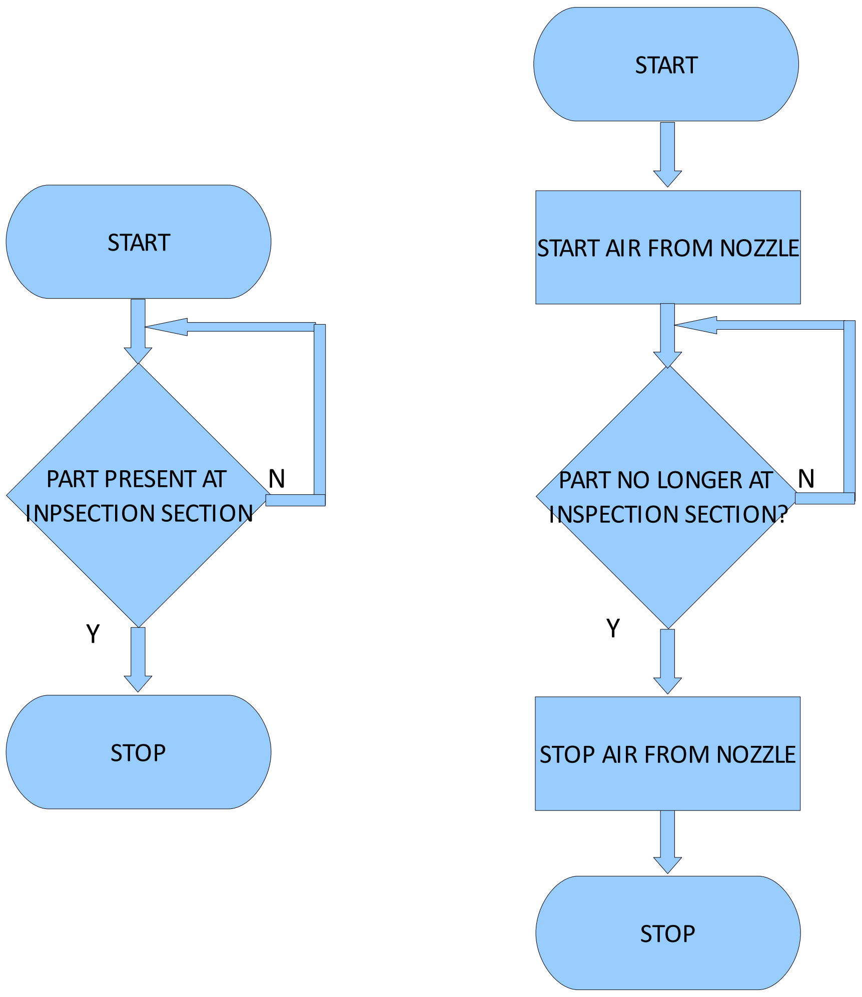
There are two methods which run simultaneously and are cyclically called by the control program. These are the hopper motor controller method and the transfer conveyor controller method. The algorithms for these methods are shown in the flowcharts in figure 4.4a. The hopper motor controller method constantly checks whether the number of parts on the first transfer conveyor is less than the minimum number of parts that should be on it (which, in this case, is 5). If it is less, the hopper motor is then run for a specified period (40 seconds, in this case) in order to increase the number of parts on the first transfer conveyor. On the other hand, the transfer conveyor controller constantly checks whether there is a part present on the inspection platform, ready to be inspected and picked in case a LOAD command is sent to the feeder. If there is no part on the inspection platform (for example, after the loaded part is picked by the robot or rejected by the singulation unit), the transfer conveyor is started so that a new part is loaded on to the platform.

The algorithms for loading and rejecting parts are shown in the flowcharts in figure 4.4b. In the case of loading of parts, it is expected that a part should already be on the inspection platform, but if that is not the case, the transfer conveyor motor is started in order to load a part. The transfer conveyor motor stops immediately when a part is loaded on to the inspection platform. On the other hand, for the rejection of parts, the air nozzle is opened to allow air jets to remove the part from the inspection platform. When the part leaves the platform, the air nozzle is closed.


START
NO. OF PARTS ON 1ST CONVEYOR LESS THAN MIN. NO. OF PARTS
N
Y
RUN HOPPER MOTOR FOR 40 SECS
STOP HOPPER MOTOR
STOP
START
PART NOT LOADED AT INSPECTION SECTION?
N
Y
RUN TRANSFER CONVEYOR MOTOR
PART REACHED INSPECTION SECTION?
N
Y
STOP TRANSFER CONVEYOR MOTOR
STOP

Hopper motor controller　　　　　Transfer conveyor controller

Figure 4.4a　Flowcharts for process flow of singulation unit


START
PART PRESENT AT INPSECTION SECTION
N
Y
STOP
START
START AIR FROM NOZZLE
PART NO LONGER AT INSPECTION SECTION?
N
Y
STOP AIR FROM NOZZLE
STOP

Loading parts Rejecting parts

Figure 4.4b Flowcharts for process flow of singulation unit

4.3 Feeder camera

The feeder camera is a Cognex DVT vision sensor. It is an “intelligent” camera and does not require a controller in order to execute its functions. The vision sensor comes with software, the Intellect software, which can be used to configure the vision sensor from a PC. The software can be used to visualize images captured by the camera, view the result of the inspection, perform general camera setup, as well as change the settings of the camera. Typically, the software is used to set up sensor tools to be used by the vision sensor for inspection and it can also be used to write scripts which are programmed to run in the vision sensor. The scripts are programmed either as background scripts or foreground scripts. Background scripts are cyclic functions that execute continuously within the vision sensor, while the foreground scripts are functions which are programmed to be triggered at various times by events or triggered externally by inputs to the camera's digital IO ports. The software is also used to calibrate the vision sensor. The calibration of the camera is done by the use of standard grids from the manufacturer. The calibration is stored within the sensor and is used to transform dimensions obtained in pixels to standard measurements such as millimeters, centimeters, etc.

Two transforms are used in this work due to the difference in sizes between the various parts to be inspected. Most of the parts measure between 10 and 60 mm in length while the assembled part is more than 100 mm in length. A standard DVT grid of 10 mm spacing was used to calibrate the camera for the smaller sized parts like the arc runner, the pigtail, etc, while a standard DVT grid of 20 mm spacing was used for the larger components such as the assembled part which will be inspected after welding. Each calibration is saved in the Intellect software as a transform and given a specific name. The transform set up using the 10 mm spacing grid was named loadTransform, while the other transform coordinate set up using the 20 mm spacing grid was named removalTransform. In order to calibrate the vision sensor, the appropriate grid is placed in the camera's field of view, and the system coordinate calibration option is selected in the Intellect software. The camera then automatically performs the necessary coordinate transform. The origin of the grid then becomes the camera coordinates' origin. Each of the parts to be inspected has a product name (and a product ID). In this case, the product names and IDs are the numbers 1 to 6 representing the moving contact, the pigtail 1, the handle frame assembly, the coil, the pigtail 2 and the arc runner respectively. The actual parts serve as the model against which each inspection is done.

Three sensor tools were set up for the vision sensor. The first tool, named filter_Tool, is used for the enhancement of the image acquired by the camera. There are a number of options available for filtering but “Morphology” with the “close” option was chosen because “this option eliminates noise inside the part and does not alter the sizes of the parts” as stated in the sensor tool guide. This filter_Tool is therefore not an inspection tool but a preprocessing tool. The advantage of pre-processing is that it improves the quality of the image acquired by the camera and compensates for poor lighting and other deficiencies. The second sensor tool used is the identification tool, named identifPart_Tool. This tool compares the image acquired by the camera with that of the model in memory and returns a PASS or a FAIL remark depending on the success or failure of the inspection. The bases of the comparison between the image and the model include blob comparison, area comparison, eccentricity, perimeter, etc. The criterion for deciding the PASS or FAIL remark of the inspection was set at 90% correlation and this is because some of the parts to be inspected appear similar from some viewing angles, so a greater correlation should be required between the image and the model before it is decided whether the inspection is successful or not. Each product has a number of models used by the identifPart_Tool to identify the part being inspected. These models represent the different collectable poses of the part. A PASS remark is returned by identifPart_Tool if there are enough similarities between the acquired image and any of the models. The matching model is contained in the “BestMatchType” property of the identifPart_Tool. The last tool is the positioning tool which returns the pick point position of the part and its angle of orientation. The positioning tool was set to reference the image identified by the identification tool. Two collectable poses were assumed in this work, and therefore, two positioning tools were used for the respective poses. The two positioning tools are named PickPos_Tool and PickPos1_Tool. The positioning tool returns the pick point position and the angle required by the feeder robot to pick up the part.

Two scripts were written to accept user input as well as coordinate the output of the sensor tools. By default, the sensor tools run concurrently but in our case, some of the tools, for example the identifPart_Tool, rely on the output of others, for example the filter_Tool, so, a script is needed to collate the output of each tool. One of the scripts written is a background script and the other is a foreground script. These scripts are included in appendix B and the syntax of the scripting language in the DVT vision sensor is similar to C. The background script runs continuously and is responsible for accepting inspection commands from the main feeding system controller, while the foreground script was configured to be triggered by the background script when it receives the INSPECT command.

Because, in our case, communication with the background script is to be done over Ethernet, two options are available for communicating the command to the camera. The first is the default command list from the manufacturer which can be understood by the vision sensor. This command list is a set of standard letters and symbols. However, there are disadvantages to the use of this list. The first is that the program becomes difficult to debug since the commands in the list are not easily comprehensible to human readers. The second is that the command list is actually for the manufacturer's internal use and a caveat was placed on its use because its functionality cannot be guaranteed. A different approach was used instead. A standard was devised to send information to the vision sensor over Ethernet using two bytes. The vision sensor reads these bytes through the background script which is always running cyclically. The interpretation of the two bytes is as follows: the first byte contains the command and the second byte contains the product ID. The use of these bytes could be extended in future, but for our work, the two bytes are sufficient. The only value of the first byte used is value 1 which was assigned to the “inspect” command. Future users of the vision sensor may find it necessary to assign other commands to other values of the first byte. The values of the second byte range from 1 to 6, and represent the product ID which is used to identify the product to be used for the inspection.

An Ethernet terminal controller using the DATALINK driver was set up so that the camera acts as a server listening for connections on port 10000. We intend to use DATALINK, which is an in-built tool by the manufacturer, to obtain inspection result output from the camera. DATALINK usually sends output through port 3247, but in this case, the output is diverted to port 10000 of the Ethernet terminal controller. The foreground script, named partScript Tool, triggers the generation of the inspection result through DATALINK when the inspection is completed. By default, the inspection result is sent out through DATALINK every time an inspection is done by any of the camera's sensor tools. However, in our case, the result is to be sent in the format described below only after the inspection is completed by all vision sensor tools. For this reason, DATALINK was configured to give out inspection reports only when a bit (the User1 bit) is triggered. The bit is triggered in the partScript foreground script when inspection is completed. This is to enable the results of the various sensor tools to be collated in an XML format. This XML format was hard-coded in the DATALINK string output format. It gives an output in the following format:

```
<MODEL>{identifPart_Tool.BestMatchType}</MODEL>

<DONE> {identifPart_Tool.ResultString} </DONE>

<X> {Pickpos_Tool.PickPoint.X}</X>

<Y> {Pickpos_Tool.PickPoint.Y}</Y>

<ANGLE> {Pickpos_Tool.PickPoint.ANGLE}</ANGLE>
```

The camera stores this as the format with which DATALINK sends out inspection results and fills in the various properties of sensor tools (for example, identifPart_Tool.ResultString) when the output message is to be sent.

4.4 Feeder robot

The feeder robot is an RTX robot and, as was stated earlier, it is without a controller. Communication between the PC controller and the robot is over a RS-232 port. The commands sent and responses received by the robot are in bytes. Depending on the type of command, command bytes may be sent in one or two transactions. In the RTX robot, a transaction is a complete cycle of sending a command and receiving a response. The interaction between the PC controller and the robot is 4 bytes long and these bytes include both the command and the response bytes. Commands are always one byte or three bytes long, while responses are also always one byte or three bytes long and for any transaction, the sum of bytes for the command and the response never exceeds 4 bytes. Therefore, if a command of length 3 bytes is sent to the robot, then a response of length one byte is received, and if a command of length one byte is sent, then there are two possibilities to the length of the response. The response could be of length one byte or three bytes. Commands of length one byte are usually those of the “supervisory” type that typically requests information from the robot controllers and these commands are usually completed in one transaction. The robot is internally controlled by two controllers called IP0 and IP1 which perform the actual control of its appendages. The particular bytes for specific commands and the expected responses received, including the number of transactions required, are given by the manufacturer in the robot's documentation files (UMI, 1986). Distances in the robot are measured in terms of encoder counts and the conversion ratios of these encoder counts to millimeters are also given by the manufacturer (UMI, 1986).

Apart from the bytes needed to control the robot, the geometry of the robot also plays an important role in the control of the robot. For the movement of the robot arm to any position, the lengths of the robot shoulder, elbow and grippers are considered and the problem is solved as a geometric problem using the laws of cosines. This is used to determine the distance (and therefore, the encoder count value) that should be sent to the robot in order for it to move to the correct position. This also applies in the case of the pitch, the roll and the gripper distance of the robot.

In the control program of the robot, a library file, similar to the RT100.lib file provided by Wane [s.a.], was developed. It was written as a dll file and it takes care of the low-level byte transactions with the robot's IP controllers. Another library file, called Interact.dll, was developed to perform the geometric calculations and convert distances given in Cartesian coordinates to encoder counts for the various motors of the robot. Three of the methods used in the control program are the moveTo(x,y,z), open(grip) and the rotateBy(angle) methods. The first method is used to move the robot arm to a given position (x,y,z), the second is used to open the grippers through a given distance “grip”, and the third method is used to rotate the grippers through a specified angle. The PicknPlace(frmX,frmY,frmZ,frmAng,toX,toY,toZ) method is used to pick up the parts from a position (frmX,frmY,frmZ) to (toX,toY,toZ). The frmAng represents the roll angle of the robot wrist.

5 Reconfigurable control of the feeding system

As discussed earlier, the feeding system consists of the singulation unit, the camera and the robot. The overall controller for the feeding subsystem is a PC. The control is distributed among the controllers of each of the subsystems mentioned above with the feeding system controller at the top of the hierarchy. Each one of the subsystems is also controlled by a PC. The communication interface between all the controllers is Ethernet. The layout is as shown in figure 5.1. In this work, the same PC is used as the controller, while the control programs use different Ethernet ports for communication.

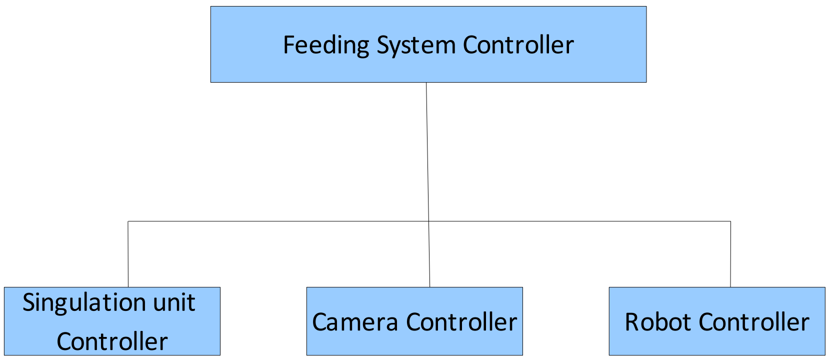


Figure 5.1 Communication layout of feeding system

Our aim in this chapter is to discuss how the control of the feeding system was implemented to make it reconfigurable. There are two methods considered in this work: the agent-based approach and the distributed approach based on the IEC 61499 standard. These two methods should interface with the controllers of each of the subsystems in the feeding system as developed in chapter 4. For this reason, a client-server approach and an XML data exchange format, both of which are discussed next, were used.

5.1 Subsystems' interaction

Since the communication between the feeding system controller and the controllers of its subsystems is over Ethernet, each of the subsystem control programs was programmed to be a server and the feeding system control program acts as a client. This is the case in the agent-based and the IEC 61499 distributed approaches. The use of socket programming makes it easy to transmit data as strings (i.e. in XML format, in this work) between the controllers. In addition, socket programming is language independent which makes the transfer of data uniform in spite of the fact that the subsystem controllers were programmed using Visual Studio, the multi-agent system is executed using JADE, and the IEC 61499 distributed control is implemented in FBDK. Socket programming also provides a seamless connection with the camera. Each controller in this work was assigned a port over which it listens for connections as a server. A list of the various ports allocated to each controller is provided in appendix C.

The XML format adopted for the exchange of data between the controllers is also given in appendix C. In order to send data from one controller to the other, the data is written as an XML document and this document is sent as a string to the output stream of the TCP/IP connection. The string is received by the other controller and it is decoded based on the standard XML format. After decoding the string, the controller can tell whether the data was intended for it or not. It can also extract the necessary information it needs in order to implement the task requested. In Visual C#, the XML document was written using XMLTextWriter and decoded using the XMLTextReader. The XML writer is used to create an XML document and this document is converted to string using a string writer. The string is then written to the TCP/IP network stream. In Java (on which the JADE platform for agents runs), the XML documents were written using a library file downloaded from the internet. The library file was developed by Jakarta ECS for creating XML documents in Java. Decoding the XML format data in Java is done by parsing the XML string using the SAX (Simple API for XML) and DOM (Document Object Model) API from W3C (i.e. the World Wide Web Consortium), which is responsible for developing standards such as XML (W3C, 2010).

5.1.1 Singulation unit control program

The singulation unit control program listens for connection from the feeding system controller on port 7210. It receives the XML string format and parses it to determine, based on the task given, whether it should LOAD or REJECT a part. If it is able to complete the task, it returns an XML string format, which contains "<DONE>true</DONE>" indicating the task was completed successfully, to the feeding system controller, else it returns a "<DONE>false</DONE>" message and the singulation unit triggers a FAULT output (e.g. if no part is loaded due to clogging).

5.1.2 Feeder camera control program

The feeder camera control program listens for connection on port 7220. It receives the XML string format and parses it to determine whether to perform an INSPECT task. The XML string also contains the part to be inspected. The control program then creates a client socket with which it connects to the Cognex camera's background script (which is listening for connection on port 3248 of the Cognex camera's IP address). The control program sends two bytes: byte value 1 for INSPECT and the second byte value to represent the ID of the part to be inspected. The Cognex camera returns whether the part was correctly identified, and it also returns the pick point position and the angle of orientation of the part. If the part was not correctly identified, the control program returns an XML string containing "<DONE>false</DONE>" to the feeding system controller. However, if the part was correctly identified, the control program carries out a coordinate transformation (discussed below), and sends the pick position and angle to the feeding system controller in XML format.

In this work, a general reference coordinate system was chosen for coordinate exchange between the feeder camera and the robot. This reference coordinate system was such that its X axis is same as the Y axis of the Cognex camera, and its Y axis is the negative of the X axis of the camera. In addition, the origin of the Cognex camera's coordinate system has an offset (223, 1190, 320), in millimeters, from the origin of the reference coordinate system. Therefore, the coordinate transformation done in the feeder camera control program is as follows:

The reference coordinate X is Cognex camera Y + X offset (i.e. 223);

The reference coordinate Y is -[Cognex camera X] + Y offset (i.e. 1190);

The reference coordinate Z is Z offset (i.e. 320).

The angle of orientation is measured clockwise from the Y axis of the camera and was set to have a value that lies between the range -180 degrees to 180 degrees.

5.1.3 Robot control program

The robot control program listens for connection on port 7230. It receives the XML string format and parses it to determine whether to perform a PICKUP task. The XML string also contains the X, Y, Z position of the part to be picked, and the angle of orientation. The control program transforms this position coordinate from the general reference coordinate system. The axes of the reference coordinate system align with the axes of the robot, but the origin of the robot's coordinate system is at an offset of (-150, 1100, 0), in millimeters, to the origin of the reference coordinate system. The control program transforms the coordinates received as follows:

The robot coordinate X is Reference X – X offset (i.e. -150);

The robot coordinate Y is Reference Y – Y offset (i.e. 1100);

The robot coordinate Z is Reference Z;

The robot roll angle is the negative of the angle of orientation (if the angle of orientation is negative) or 180 – angle of orientation (if the angle of orientation is positive).

The roll angle is set to be perpendicular to the angle of orientation of the part, but since the roll angle is measured counterclockwise from the robot's Y axis (which is perpendicular to the Cognex camera's Y axis), the roll angle is thus 180 – the angle of orientation. In addition, a given roll angle θ is the same as the roll angle 180 + θ.

5.2 Agent-based control

Multi-agent systems are one of the means of achieving reconfigurable control. One of the ways of making the feeding system reconfigurable is to ensure the independence of the subsystems within it, i.e. the singulation unit, the camera and the robot. Hence, the feeding system can be seen as a holonic system in which the individual subsystems are independent. Holons can be implemented using agents.

5.2.1 Holons in the feeding system

According to the methodologies for holonic systems, such as PROSA, ADACOR, etc, the physical components or machines form a class of holons. Therefore, the singulation unit, the camera and the robot are holons and they are referred to as the resource holons in PROSA and operational holons in ADACOR. In this work, the singulation unit holon, the camera holon and the robot holon are each implemented as feederAgent, cameraAgent and robotAgent agent classes in the multi-agent system. Also, there is the taskAgent holon which performs the task of the product holon and the task holon, as specified in PROSA and ADACOR respectively. Finally, there is the controllerAgent holon. This holon maintains the other holons and is referred to as the staff holon in PROSA and the supervisor holon in ADACOR. From the foregoing, five holons, and therefore five agents, were identified: the supervisor holon, the singulation unit holon, the camera holon, the robot holon and the task holon, which are implemented as the controllerAgent, the feederAgent, the cameraAgent, the robotAgent and the taskAgent agent classes respectively. These agents were implemented on the JADE platform (JADE, 2010).

As mentioned in chapter 2, JADE is one of the programming languages for implementation of multi-agent systems. One of the reasons for choosing JADE in this work is that it is a well developed and researched language. Since 1998 when it was developed by the Telecom Italia (formerly CSELT), it has constantly been updated and the latest version, JADE 3.7, was released in June 2009. Another reason for the choice of JADE is the large number of resources and support available for the language by other developers. Furthermore, the language is FIPA compliant and its communication framework complies with the agent communication language (ACL) of FIPA. Also, in JADE, different interaction protocols already exist within the software framework so that there is no need to program an interaction protocol. Examples of the protocols, which are used in this work for bargaining between the agents, are the FIPA-Request and the FIPA-Contract-Net protocols. In JADE, the different actions which an agent is able to perform are programmed as behaviors.

In order for agents to communicate meaningfully, it is necessary to develop an ontology. An ontology was developed for this case study using the Protege software and JADE ontology bean generator. The ontology consists of concepts, actions and predicates. Concepts are entities with structure and properties, and are typically nouns. Actions, on the other hand, represent the agent actions and are typically verbs. Predicates indicate some form of relationship between concepts. The actions used in the multi-agent system include Inspect, Load, PicknPlace and Reject. The concepts used include Controller, Duration, Part and Position, while the predicates used include hasControllerName, isPosition and lostEffector. This ontology was automatically generated by the JADE ontology bean generator from Protege. The generated ontology files were then programmed.

Before the discussion of the behaviours of each of the agents, two actions common to all the agents will be mentioned. The first is the registration of the agents in the yellow pages of the JADE platform. JADE has a yellow-pages function which allows agents to publish their services so that they can be located by other agents. All the agents used in this multi-agent system publish their services as part of their setup() method. They also include a service description by which they can be found, for example, the robotAgent gives its service description as “robot”. This implies that when a search criterion is done using “robot”, all robot agents with that service description will be located. The second action common to all the agents is the “deregistration” of the agents from the yellow-pages service when the agents are terminated. This is included in the takeDown() method of the agents and is called when an agent is to be terminated. This action is also provided by the supervisor holon as a behaviour.

5.2.2 Supervisor holon

The supervisor holon is implemented as the controller agent. The supervisor holon is responsible for creating the agents for the other controllers and performs the clean up when any agent is terminated. The controller agent is the first agent that is created and remains in the main agent container when the agent platform starts up. The controller agent manages the existent controllers within the multi-agent system, and it informs the JADE AMS (agent management system) of which agents are to be created or terminated. The behaviours (or actions) of this agent are called createController and killController.

The createController behaviour is a OneShotBehaviour, which in JADE, is a behaviour that runs only once. It is activated when the controller agent receives any messages from a new controller connected to the multi-agent system. The controller agent requests the JADE AMS to create an agent for the new controller. In the case study of the feeding system, the controller agent creates the feederAgent, the cameraAgent, the robotAgent and the taskAgent agents upon system start up. It does this by adding new createController behaviours with each of the agent names in the program thus:

addBehaviour(new createController(feederAgtName, feederAgtClass));

addBehaviour(new createController(cameraAgtName, cameraAgtClass));

addBehaviour(new createController(robotAgtName, robotAgtClass));

addBehaviour(new createController(taskAgtName, taskAgtClass));

The other behaviour, the killController, is a cyclic behaviour. In JADE, this means that the behaviour runs continuously throughout the life of the agent. The purpose of this behaviour is to inform the AMS to deregister any agent, and its services, that has been terminated. This deregisteration includes removing the services of the terminated agent as well as reducing the number of agents of that kind within the system. This behaviour is added thus: addBehaviour(new agentKiller());

5.2.3 Singulation unit holon

The singulation unit holon is implemented as the feeder agent. The feeder agent performs two actions: loading parts and rejecting parts. It, therefore, has two behaviours which represent these actions. These behaviours are called loadPart and rejectPart, and are both cyclic behaviours. Each of these behaviours cyclically scans all messages received by the feeder agent. The loadPart behaviour looks for “load” messages, while the rejectPart behaviour looks for “reject”. These behaviours are implemented to respond to FIPA-Requests and FIPA-Contract-Net messages. In the case of FIPA-Requests messages, the relevant behaviour executes the task immediately and responds with a FIPA-Inform or FIPA-Failure depending on whether it performs the task successfully or not. In the case of FIPA-Contract-Net call for proposal, the relevant behaviour responds by sending a proposal of its duration time which the contract-net initiator may accept or not. If the proposal is accepted, then the task is executed. The task agent used in the case study of the feeding system, which is responsible for initiating request messages, uses the FIPA-Request. The behaviours of the feeder agent are added thus: addBehaviour(new loadPart()); for loading parts, and addBehaviour(new rejectPart()); for rejecting parts.

5.2.4 Camera holon

The camera holon is implemented as the camera agent. The camera agent performs only one action: inspecting parts. It, therefore, has just one behaviour: the inspectPart behaviour. This behaviour is a cyclic behaviour. The inspectPart behaviour continuously scans all messages received by the camera agent for “inspect” messages. The response of the behaviour to FIPA ACL messages is similar to that described above in the case of the feeder agent. The behaviour of the camera agent is added thus: addBehaviour(new inspectPart());.

5.2.5 Robot holon

The robot holon is implemented as the robot agent. The robot agent performs only one action, which is to pick up parts and place them in the fixtures on the pallets. This action is represented as a cyclic behaviour which is called pickPart behaviour. The pickPart behaviour cyclically scans all messages received by the robot agent for “PicknPlace” messages. The response of the pickPart behaviour to FIPA ACL messages is similar to that described for the case of the feeder agent. The behaviour of the robot agent is added thus: addBehaviour(new pick());.

5.2.6 Task holon

The task holon is implemented as the task agent. The task holon generates the task to be done, and is responsible for sending request messages to other agents. It carries out the steps necessary for the feeding system to load a part by coordinating the activities of the other agents, and it takes decisions based on the outcome of each of the tasks performed by the other agents. It determines the procedure and sequence of the job to be done. When a part is required to be loaded onto a pallet, it sends a message to the singulation unit holon requesting it to load a part. If the singulation unit is not able to load the part, a fault message is printed by the task holon, but if the task holon is informed that the part has been successfully loaded by the singulation unit holon, it sends another message to the camera holon to inspect the part. If the camera holon informs the task holon that the inspection was successfully done (and the part's pick position is supplied), the task holon sends a message to the robot to pick the part, but if the inspection response message was a failure message, the task holon sends a message to the singulation unit holon to reject the part. If the robot holon returns a failure message, the singulation unit holon is requested to reject the part, but if the robot successfully picks the part, the camera holon is requested to confirm by inspection whether the part has been picked. If no part is detected, the task is completed, but if a part is detected, the singulation unit holon is requested to reject it. The behaviour has handleInform and handleFailure methods for each type of message returned to it. These steps are implemented in the task agent as a stepped behaviour called the nextJob behaviour. A variable called step is defined in the behaviour and the value is only increased when the current task is successful and then, the behaviour proceeds to the next step. This behaviour is introduced as addBehaviour(new nextJob());.

The task agent also has three additional behaviours: the feederAgentList, the cameraAgentList and the robotAgentList behaviours. The agent uses these behaviours to request the list of all feeder agents, camera agents and robot agents in the multi-agent system. The behaviours sends these requests to the JADE DF (Directory Facilitator), which uses the service descriptions (which in our case are feeder, camera and robot respectively) of the agents to locate them in the yellow pages of the JADE platform.

5.3 Distributed control based on IEC 61499

The IEC 61499 standard is another methodology for the distributed control of holonic systems aimed at system reconfigurability. This method is also applied to the control of the feeding system. In the distributed control method, the controllers are programmed using function blocks and the controller of each subsystem is configured as an independent device. Therefore, for the feeding system, the singulation unit, the camera and the robot controllers are configured as devices in the function block application. Each of these devices has function blocks suitable for its operation mapped into it. The IEC 61499 function block model of the feeding system is developed using the FBDK from Holobloc Inc. The FBDK is used because it has the longest history of all the software development kits available for the IEC 61499 standard and it provides services which range from the development of function blocks to the execution of the function block application. In addition, FBDK is still being constantly used and developed, and there is an active group of users of FBDK that participate in an FBDK googlegroup where ideas are shared on the software with its developers.

From the discussion above, three devices have been identified respectively for the singulation unit, the camera and the robot. These were named as the Feeder device, the Camera device and the Robot device in the function block application model. In addition, there are the IN_DEV1 and OUT_DEV1 devices, which are HMI devices set up for diagnostic purposes and they will be discussed further in section 5.3.2. A full description of each function block and the distributed application as mapped into the various devices is given in appendix D. Similar to the case of the agent-based control method, the controller devices have to communicate with the subsystem controllers developed in chapter 4. In the distributed control method, socket programming and the XML data format are also used, except that they are done with the use of function blocks. FBDK provides a standard function block for client-server communication over Ethernet. The creation of XML documents and parsing of XML strings were done using the XML encoder and parser function blocks which we discuss next.

5.3.1 XML encoder and parser function blocks

The XML encoder and parser function blocks were developed to enable the function blocks in the controller devices, i.e. the feeder device, the camera device and the robot device, to communicate with the subsystems' controllers, i.e. the control programs of the singulation unit, the camera and the robot developed in chapter 4. The XML encoder function blocks are used to create XML documents that are sent as strings to the subsystems' controllers, while the XML parser function blocks are used to parse XML strings received from the subsystems' controllers.

There are two types of XML encoder function blocks developed in this work and they are shown in figure 5.2 below. The first of them, XML_ENCODER_1, is used

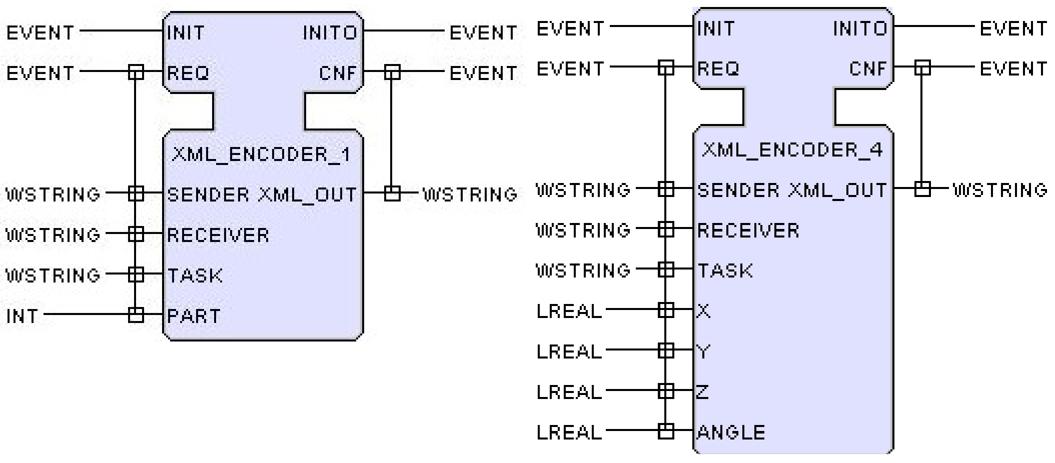


Figure 5.2 XML encoder function blocks

to create XML string formats that are sent to the singulation unit and the camera control programs. The second function block, XML_ENCODER_4, is used to create XML string formats that are sent to the robot control program. The reason for these different function blocks is the XML standard used in this work. The XML format for the control programs of the singulation unit and the camera only require the task to be executed and the part to be loaded or inspected, while the robot control program needs to be supplied the X, Y and Z positions (relative to the origin of the reference coordinate system) of the part to be picked, along with the angle of orientation. Apart from the difference in the number and the type of inputs required, both encoder function blocks work in the same way. Figure 5.3 shows the execution control chart (ECC) of the function blocks. The REQ algorithm shown also uses the Jakarta ECS library file to create XML document. The .java files which FBDK generates for each function block were modified so that the Jakarta ECS library file can be used to generate XML documents. The modified .java files were then recompiled and used to replace the FBDK compiled .class files.

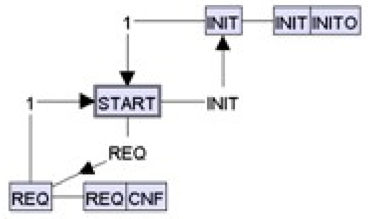


Figure 5.3 ECC of XML encoder function blocks

A similar procedure was followed for the XML parser function blocks shown in figure 5.4. There are also two parser function blocks: the XML_PARSER_0, which is used to parse XML strings received from the singulation unit and the robot control programs, and the XML_PARSER_4, which is used to parse XML strings received from the camera control program.

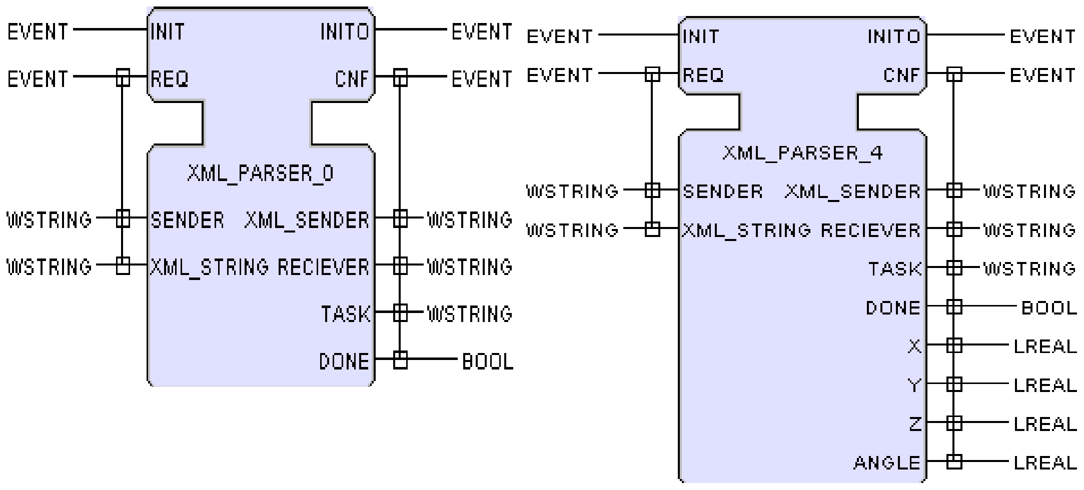


Figure 5.4 XML parser function blocks

The camera function block uses XML_PARSER_4 because it returns four values representing the pick position and angle of the inspected part, while the singulation unit and the robot function blocks use XML_PARSER_0 because the XML string to be parsed only returns one value which represents whether the task was successfully done or not.

5.3.2 IN_DEV1 and OUT_DEV1 HMI devices

The IN_DEV1 and the OUT_DEV1 were set up as HMI devices. In FBDK, HMI devices have panel resources, which enable the display of input control buttons and output display respectively. Figure 5.5 shows the graphic displays of the IN_DEV1 and the OUT_DEV1 devices. IN_DEV1 is used to accept input from the user. It displays a drop-down list from which the user can make choices. As shown in the figure, the two choices available are LOAD and REJECT part. There is a text box which accepts user input of the type of part to be loaded, and a control button marked GO which starts the LOAD or REJECT action. OUT_DEV1, on the other hand, is only used for diagnostic purposes. It reports what task is currently in progress and reports whether it was successfully done or not. It receives the updates on the progress of work from each of the other devices and updates the display based on the progress report.

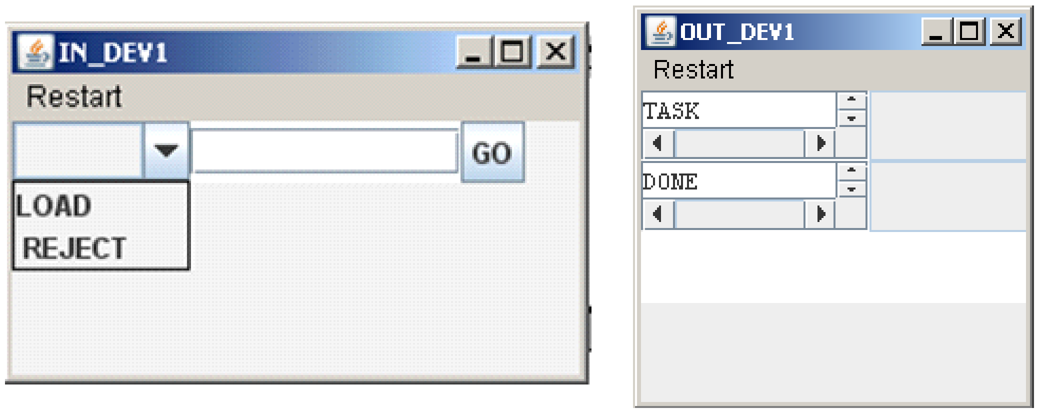


Figure 5.5 IN_DEV1 and OUT_DEV1 user interfaces

The function blocks used in the IN_DEV1 and OUT_DEV1 are given in appendix D. All the function blocks are standard function blocks provided in FBDK except HMI_EXIT function block. HMI_EXIT is programmed to read the choice of the user and it generates a LOAD or REJECT event depending on the choice of the user. This event is communicated to the Feeder device through a publisher function block.

5.3.3 Feeder device

The Feeder device contains the function blocks which control the singulation unit. It contains a subscriber block, a composite function block called the FB_FEEDER, and two publisher blocks. The Feeder device receives events and data from other devices through the SUB1 subscriber block and it sends events and data to the Camera device for inspection through one of the publisher blocks, the PUB2 publisher block. It also sends event updates to the OUT_DEV1 device through the other publisher block, the PUB5 publisher block.

The main control function takes place within the FB_FEEDER function block. The FB_FEEDER function block, apart from the customary INIT and INITO events, has two event inputs: the LOAD and the REJECT events, which represent the tasks that the singulation unit executes. It also has two event outputs: the CNF and the INSPECT events. The CNF event confirms that the task assigned to the singulation unit has been completed and a Boolean variable associated with the event indicates whether the task was successful or not. The INSPECT event is an event which is sent to the Camera device, and it is only generated if the input event to the FB_FEEDER was a LOAD event and the loading was completed successfully.

The FB_FEEDER function block consists of the FEEDER_ENTRY, the XML_ENCODER_1, the CLIENT_1, the XML_PARSER_0 and the E_SPLIT (which is the FEEDER_EXIT) function blocks. The function block network in the FB_FEEDER is shown in figure 5.6. The function of each of these function blocks is as follows: the FEEDER_ENTRY function block determines what task is to be performed by the singulation unit. The task to be performed is received as an input event from the IN_DEV1 device. FEEDER_ENTRY then sets the TASK string variable to either "LOAD" or "REJECT" and sends this to the XML_ENCODER_1 function block. XML_ENCODER_1 also receives the type of part to be loaded, stored in the PART variable, from the IN_DEV1 device and with this, it creates the XML data format for the string that will be sent to the control program of the singulation unit. If the event is a REJECT event, then there is no need for a PART variable.

The XML string output is sent to the control program of the singulation unit through the CLIENT_1 function block, which serves as a client over Ethernet. The response from the singulation unit's control program is also obtained through the CLIENT_1 function block. The response is in XML format and has to be parsed so it is sent from the CLIENT_1 to the XML_PARSER_0 function block. XML_PARSER_0 parses the XML string to determine whether the task was completed successfully or not. In the case of a REJECT event, there is no need to generate any further events, but in the case of a LOAD event, a CNF event is generated by the XML_PARSER_0 which is split into two by the E_SPLIT function block. One of the events is sent to the OUT_DEV1 device to update the display output, while the other is an INSPECT event which is sent to the Camera device for it to inspect the loaded part.

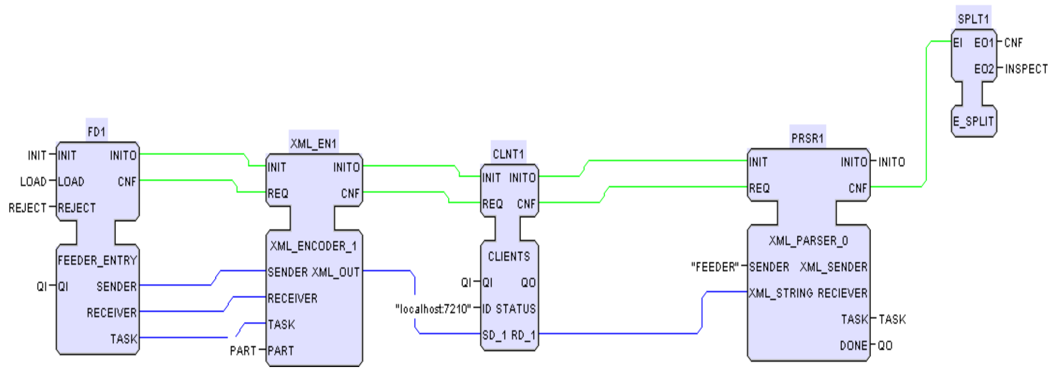


Figure 5.6	FB_FEEDER composite function block

5.3.4 Camera device

The Camera device contains the function blocks which control the camera. It contains a subscriber block, a composite function block called the FB_CAMERA and three publisher blocks. The Camera device receives events and data from other devices through the SUB2 subscriber block. The only type of event the Camera device responds to is the INSPECT event and it sends events and data to three other devices. It uses the PUB3 publisher block to send PICKUP events to the Robot device when a part is successfully inspected by the camera, and it uses the PUB4 publisher block to send REJECT events to the Feeder device in cases where there is an inspection failure. It also sends event updates to the OUT_DEV1 device through the PUB6 publisher block.

The main control function takes place within the FB_CAMERA function block. The only task performed by the Camera device is to inspect so the FB_CAMERA function block has only one event input, the INSPECT event, apart from the INIT event. It also has four event outputs, which include the INITO, the CNF, the PICKUP and the REJECT output events. The CNF event is generated when the inspection task is completed by the Camera device and a Boolean variable associated with the event indicates whether the task was successful or not. This event is sent to the OUT_DEV1 device for it to update the display output. Depending on the outcome of the inspection task, FB_CAMERA generates either a PICKUP event which is sent to the Robot device (along with the pick position and angle of the part) if the inspection was successful, or a REJECT event which is sent to the Feeder device if the inspection was not successful.

The FB_CAMERA function block consists of the CAMERA_ENTRY, the XML_ENCODER_1, the CLIENT_1, the XML_PARSER_4, the E_SPLIT and the

CAMERA_EXIT function blocks. The function block network in the FB_CAMERA is shown in figure 5.7. The function of each of these function blocks is as follows: the CAMERA_ENTRY function block receives an INSPECT input event from the Feeder device. It then sets the TASK string variable to "INSPECT" and sends this to the XML_ENCODER_1 function block. XML_ENCODER_1 also receives the type of part to be inspected, stored in the PART variable, from the Feeder device and with this, it creates the XML data format for the string that will be sent to the control program of the camera.

The XML string output is sent to the control program of the camera through the CLIENT_1 function block, which serves as a client over Ethernet. The response from the camera's control program is also obtained through the CLIENT_1 function block. The response is in XML format and has to be parsed, so it is sent from the CLIENT_1 to the XML_PARSER_4 function block. XML_PARSER_4 is used in this case because the string from the camera's control program will contain the pick position (X, Y, Z) and angle of orientation of the part. XML_PARSER_4 parses the XML string to determine whether the inspection was completed successfully or not, and if it was successful, it parsers further for the X, Y, Z and ANGLE values of the pick position. A CNF event is generated by the XML_PARSER_4 and this is split into two events by the E_SPLIT function block. One of the events is sent to the OUT_DEV1 device to update the display output, while the other is sent to the CAMERA_EXIT function block. CAMERA_EXIT also receives the Boolean variable DONE, which indicates whether the inspection was successfully carried out or not, from XML_PARSER_4. If DONE is true, then CAMERA_EXIT generates a PICKUP event which is sent to the Robot device, but if DONE is false, then CAMERA_EXIT generates a REJECT event instead, which is sent to the Feeder device for it to reject the part.

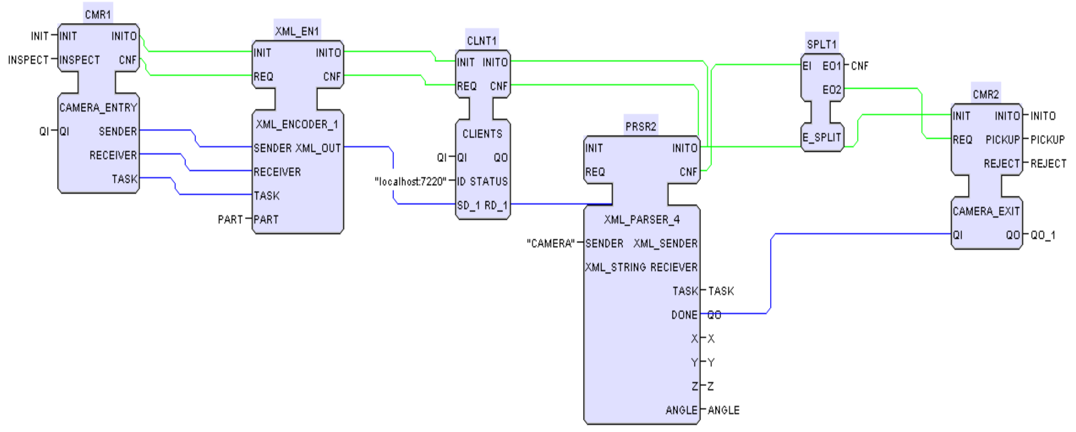


Figure 5.7 FB_CAMERA composite function block

5.3.5 Robot device

The Robot device contains the function blocks which control the robot. It contains a subscriber block, a composite function block called the FB_ROBOT and three publisher blocks. The Robot device receives events and data from other devices through the SUB3 subscriber block. The only type of event the Robot device responds to is the PICKUP event and it sends events and data to three other devices. It uses the PUB8 publisher block to send REJECT events to the Feeder device when the robot is unable to pick up a part, the PUB9 publisher block to send INSPECT events to the Camera device after it has picked the part, and it also sends event updates to the OUT_DEV1 device through the PUB7 publisher block.

The main control function takes place within the FB_ROBOT function block. The only task performed by the Robot device is the pickup task so the FB_ROBOT function block has only one event input, the PICKUP event, apart from the INIT event. It also has three event outputs: the INITO, the CNF and the REJECT output events. The CNF event is generated when the robot has completed the pickup task and a Boolean variable associated with this event indicates whether the task was completed successfully or not. This event is sent to the OUT_DEV1 device for it to update the display output. Depending on the outcome of the pickup task, FB_ROBOT generates a REJECT event, which is sent to the Feeder device, only if the part was not successful picked, otherwise the CNF event generated is sent to the Camera device to inspect whether the part has truly been picked.

The FB_ROBOT function block consists of the ROBOT_ENTRY, the XML_ENCODER_4, the CLIENT_1, the XML_PARSER_0, the E_SPLIT and the ROBOT_EXIT function blocks. The function block network in the FB_ROBOT is shown in figure 5.8. The function of each of these function blocks is as follows: the ROBOT_ENTRY function block receives a PICKUP input event from the Camera device. It then sets the TASK string variable to “PICKUP” and sends this to the XML_ENCODER_4 function block. XML_ENCODER_4 is used in this case because the pick position X, Y, Z and ANGLE of the part will be included in the XML data string. XML_ENCODER_4 receives the pick position from the Camera device, and then creates the XML data format for the string that will be sent to the control program of the robot.

The XML string output is sent to the control program of the robot through the CLIENT_1 function block, which serves as a client over Ethernet. The response from the robot's control program is also obtained through the CLIENT_1 function block. The response is in XML format and has to be parsed, so it is sent from the CLIENT_1 to the XML_PARSER_0 function block. XML_PARSER_0 parses the XML string to determine whether the pickup of the part was completed successfully or not. A CNF event is generated by the XML_PARSER_0 and this is split into two events by the E_SPLIT function block. One of the events is sent to the Camera device and the OUT_DEV1 device to update the display output, while the other is sent to the ROBOT_EXIT function block. ROBOT_EXIT also receives the Boolean variable DONE, which indicates whether the pickup task was successfully carried out or not, from XML_PARSER_0. If DONE is true, then the Camera device proceeds to inspect, while no further event is generated in ROBOT_EXIT, but if DONE is false, i.e. pickup was not successfully done by the robot, then ROBOT_EXIT generates a REJECT event which is sent to the Feeder device for it to reject the part.

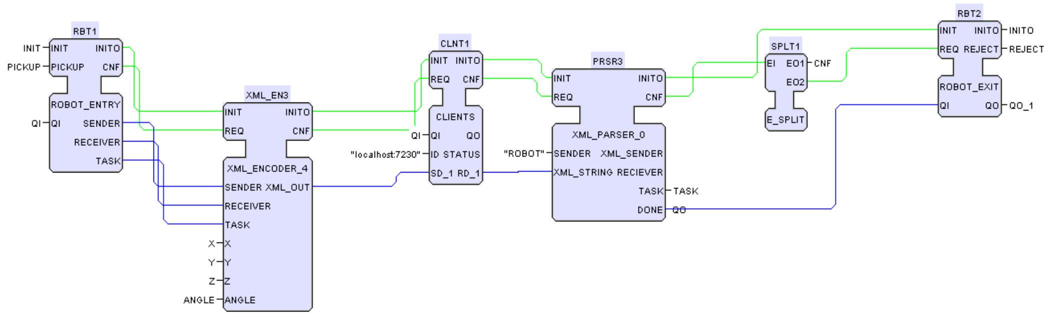


Figure 5.8 FB_ROBOT composite function block

6 Comparison between controllers and control methods

The control of the pallet magazine was done using a PC and a PLC as controllers, and two different control methods have been used for reconfigurable control of the feeding system. We now proceed to compare the two controllers used and to compare the two control methods used.

6.1 Comparison between PC and PLC

The PC used in this work has an Intel Pentium 4 CPU with a processor speed of 3.00 GHz. It also has 0.99GB of RAM, and runs Microsoft Windows XP Professional Version 2002 Service Pack 3. The hardware modules used with the PC are the Applicom card, PCIE1500PFB, for PROFIBUS communication, and the Eagle uDAQ Lite device for reading from and writing to the digital ports. On the other hand, the PLC used has a SITOP 24V, 5A power supply with a Siemens CPU 315-2 PN/DP which is capable of PROFINET and PROFIBUS communication. The PLC signal modules are DI16XDC24V, DO16XDC24V/0.5A and DI8/DO8XDC24V/0.5A. The software used to configure and program the PLC is the SIMATIC Manager STEP 7 V5.4 + SP5. The comparison between the PC and the PLC as controllers is done using the criteria for measuring the quality of a controller given by Lewis (1998). These criteria include: capability, availability, usability and adaptability. Cost will be considered as an additional criterion. The comparison made below is strictly based on what was experienced in this work.

6.1.1 Capability

Capability is the extent to which a system can perform its intended design function (Lewis, 1998). It is influenced by the following factors:

- ***Responsiveness****:* this is the time required for the system to produce appropriate responses to a specified combination of external events (Lewis, 1998).

  The PLC is more responsive than the PC, since it takes a shorter time to respond to external events than the PC. The test was to consider the time taken by each controller to raise the conveyor assembly of the pallet magazine and then close the jaws. The process took over 2 seconds for the PC controller while it took less than a second for the PLC controller.

- ***Processing capacity****:* this is the extent to which the system can meet scheduling deadlines under specified sets of conditions (Lewis, 1998).

  The processing capacity of the PLC is higher than that of the PC. As can be seen from the test mentioned above, the PLC is capable of meeting stricter deadlines than the PC.

- ***Storage capacity****:* this is the extent to which the system can retain in memory all the required programs and data under specified sets of conditions (Lewis, 1998).

  The PC has a higher storage capacity than the PLC. While the PLC has adequate memory to store its data and programs, it is without doubt that the PC has greater provision for memory to store data and programs. The memory capacity of the PLC is typically between 128KB and 2MB, while the PC may have a memory size of up to 250GB. The PLC generally stores only one program at a time, which is the program it runs when it is put in run mode. If a different program is to be run on the PLC, then that program has to be uploaded separately. On the other hand, for the PC, it can store many programs simultaneously and any one of the programs can be run at any time (and concurrently too).

6.1.2 Availability

Availability is the proportion of time in the life of a system when it is available for its intended function (Lewis, 1998). It is influenced by the following factors:

- ***Reliability***: this is the ability of the system to continue to perform all its intended functions over a specified period of time and range of conditions (i.e. the inverse of the mean time between failures) (Lewis, 1998).

  The PLC is more reliable than the PC. Provided the PLC is correctly configured, it is less prone to failures than the PC. An example of the PC's abrupt failure is that sometimes, while running, the PC suddenly gives an OS (operating system) Load Locker error when it accesses the data acquisition card's dll methods. Therefore, since the mean time between failures for the PLC is lower than for the PC, then the PLC's reliability, which is the inverse of the mean time, is higher than the PC's reliability.

- ***Maintainability****:* this is the ease with which the system can be restored to full capability after the occurrence of one or more faults from a specified set. It is the inverse of the mean time of repair (Lewis, 1998).

  The PLC is more maintainable than the PC. This is because the PLC software makes provision for diagnosis of faults. It is able to report both hardware and software faults. This makes identification of faults faster than in the case of the PC. Repair may then be carried out by replacing faulty hardware components or correcting wrong program logic. Moreso in the PC, the debug tools cannot be used to debug dll files obtained from other sources. Therefore, the mean time of repair is shorter for the PLC, which implies that maintainability is higher for the PLC than the PC.

- ***Integrity**:* this is the degree to which the system can continue to perform all its intended functions over a specified range of threats, including both unintended user actions and intentionally hostile actions (Lewis, 1998).

  Threats may be considered to be hardware-related or software-related. The PC has a higher integrity than the PLC for some hardware-related threats, while the PLC has a higher integrity for some software-related threats. For hardware-related threats, this is demonstrated when the PLC and the PC programs are run without connecting the PROFIBUS cable with which they communicate with the drive inverter used in the pallet magazine. In the case of the PLC, it immediately generates a Bus Fault and a Signal Fault and goes into STOP mode. Whereas for the PC, the error is reported, and the PC program can then decide whether it is safe to continue other operations while retrying to establish communication via the Profibus, or to halt program execution. For software-related threats, while the PC may be susceptible to a virus attack, the PLC is much less prone to such virus attacks.

6.1.3 Usability

Usability is the ease with which a specified set of users can acquire and exercise the ability to interact with the system in order to perform its intended functions (Lewis, 1998). For reconfigurable systems, it is necessary to extend the users in Lewis' definition of usability to include not only end-users, but also, system developers. Usability is influenced by the following factors:

- ***Entry requirements**:* this is the amount of formal and informal training required before the user can learn to interact with the system (Lewis, 1998).

  Entry requirements for the end-users of the PLC are slightly less than those of the PC. This is because less training is required to use the PLC than is required to use the PC. No special training is required to use a PLC controlled system other than switching buttons on and off. For the PC, a user needs to learn to use the keyboard and mouse. However, the entry requirements for system developers of the PC and the PLC are the same.

- ***Learning requirements**:* this is the training required for a user [who has met] a specified set of entry requirements (Lewis, 1998).

  The learning requirements for the developer who will use the system to develop a control program are less for the PLC than those for the PC. Part of the training required for the developer of either type of controller is the use of the programming languages. The amount of training required to be able to use PLC programming languages, such as Ladder Logic, is less than that required to use PC programming languages, such as C#.

- ***User productivity**:* this is the number of system-related operations per unit time which can be performed by a user with a specified level of training and experience (Lewis, 1998).

  A definite verdict on the user productivity is difficult to give. This is because there are tasks which require more system-related operations in the PLC than in the PC, and other tasks which require more system-related operations in the PC than in the PLC. A typical example of a system task is the case of adding new signal (digital input and output) modules to the controller. For the PC, all this requires may be plugging the module into a USB port as in the case of the Eagle USB card without making any system modifications, while for the PLC, the system must be stopped and the hardware configuration must be modified in the PLC software and reloaded after the signal module has been added. On the other hand, if the module to be added is a field bus module such as the Applicom card for the PC, the system must be shut down. The Applicom card is then slotted into the PCI bus of the PC and the card's software is installed, while for the PLC, the amount of operations required to install interface modules is exactly the same as the case of adding signal modules.

- ***Congeniality**:* this is how “user-friendly” the system is (Lewis, 1998).

  The PC is more congenial than the PLC. This is because the PC program interface can be made more user-friendly for the final user than the PLC. The PC usually has a graphic user interface, while the PLC has no such interface. However, this is provided the use of an HMI screen with the PLC is not considered.

6.1.4 Adaptability

Adaptability is the ease with which the system may be changed in various ways from its initial intended functions (Lewis, 1998). This is the criterion that is most related to reconfigurability. Adaptability is influenced by the following factors:

- ***Improvability**:* this is the ease with which system attributes can be upgraded without affecting system functionality (Lewis, 1998).

  The PC is more improvable than the PLC. This is because, in the PC, different components, software and software updates can be installed on the PC to improve it without affecting its functionality. In the case of the PLC, this is more difficult to do because even moving from one PLC series range to another one by the same manufacturer may lead to program incompatibility thereby necessitating a change of software. An example is the case of the Siemens S-200 and S-300 series PLC which use different software and have incompatible programs.

- ***Extensibility**:* this is the ease with which new functionality can be added to the system (Lewis, 1998).

  The PC is more extensible than the PLC. This is similar to improvability, and it is easier to add new functionality to a PC than to a PLC.

- ***Portability**:* this is the ease with which system functionality can be moved from one system to another (Lewis, 1998).

  PC programs are more portable than PLC programs. A PC's functionality may be moved and accessed across different PCs and PC software from different manufacturers. For example, the Applicom card which was used as the PC fieldbus module can be used even in PCs from different manufactures. However, in the case of the PLC, such portability is impossible or very difficult. For example, the fieldbus functionality cannot be moved easily from a Siemens S-300 PLC to a Siemens S-1200 PLC because the S-1200 PLC cannot be used as a PROFIBUS master.

- ***Reusability**:* this is the ease with which the functional capabilities of an existing software element can be used in a new or different system (Lewis, 1998).

  PC software is more reusable than PLC software. Existing programs in a PC are easily transferable and reusable in different PCs while for the PLC, it is likely that existing programs are only transferable and reusable between exactly the same PLCs. In fact, the number of hardware modules in both PLCs must be the same and arranged in the same order else the program from one PLC cannot be reused in the other.

In summary, for the case study considered here, there are some PLC attributes which give it some advantages over the PC, while there are other PC attributes which give it some advantages over the PLC. The choice of which one to use in a given application will depend on which of the controller attributes the user considers as relatively more important than others. However, the PC, because of its higher adaptability, is more suitable for application in reconfigurable systems.

6.1.5 Cost

In addition to the attributes discussed above, we mention cost. The cost is considered in terms of the monetary value of the controller and the time taken to get the various components of the controller to work. The monetary value of the PLC, which includes the cost of the DIN rail, the CPU, the signal modules, the SITOP power supply, the front connectors, and the 512KB memory card, is R36828.12. The cost of the PC, which includes the cost of the CPU, the monitor, the keyboard, the mouse, the Applicom card, and the Eagle uDAQ Lite device, is R29020.12. This shows that the PC is marginally cheaper than the PLC.

With regards to the time taken to get the components to work, this is subjective and the following is based entirely on our own experience. It took a shorter time to get the PLC to work than it took for the PC. The PLC modules are from the same manufacturer and are closely integrated, so getting the various modules to work only took about 2 weeks which is the period it took to learn and become familiar with PLCs and PLC programming. The time to get the various PC components to work was much longer. It includes the time taken to read through the manuals of the various components and avenues to learn how to use these components (particularly the Applicom card) were not available. In all, it took about 6 weeks to acquire the requisite skill to use all the components of the PC.

6.2 Comparison between agent-based control and distributed control based on IEC 61499

The comparison between the agent-based control method and the IEC 61499 distributed control method is done using the characteristics of reconfigurable systems given by ElMaraghy (2006) as the criteria. These characteristics include: modularity, integrability, convertibility, diagnosability, customization and scalability. Modularity, integrability and diagnosability reduce the reconfiguration time and effort, while customization and convertibility reduce cost. In addition, the fault tolerance of both control methods is considered.

6.2.1 Modularity

The IEC 61499 distributed control system is more modular than the agent-based control system. This is because every device in the IEC 61499 distributed control system is entirely independent and only sends to and receives events and data from other devices. On the other hand, in an agent-based system, there is always a “host” system on which the main container runs. Therefore, in one way or the other, all the agents are connected to this main host system and any threat to the host affects the entire multi-agent system. The IEC 61499 system does not have any such "host". In the feeding system agent-based control, the Controller Agent resides in the host.

6.2.2 Integrability

This is a measure of a system based on the performance of its components and interfaces of both software and machine hardware modules (Koren et al, 1999). It shows whether the system is ready for both integration and future introduction of new technology (ElMaraghy, 2006).

The agent-based system is more integrable than the IEC 61499 control system. This is because, in the agent-based control system, any agent can join the multi-agent system (if the agent and the system are FIPA compliant, and they have compatible ontology) even at run time, while in the IEC 61499 control system, the new system has to be explicitly added to the system configuration. Furthermore, in the agent-based control system, once a new agent joins the multi-agent system, it can register and publish its services so that any other agent within the multi-agent system that requires those services can locate the new agent in the yellow-pages service. A message requesting the new agent to perform any service may then be sent to it. However, in the IEC 61499 distributed control system, it is necessary that the new system or device is explicitly added to the system configuration and included in the subscriber list by adding a subscriber function block with the appropriate address (plus a publisher function block or a client/server connection if that is necessary).

In the feeding system agent-based control, if an additional feeder is to be added to the system, the new Feeder Agent is created and is registered in the main container and publishes its services so that it can be located by the Task Agent in the yellow-pages service. The Task Agent will now see two feeder agents and it can choose between them using the contract-net protocol. In the case of the IEC 61499 control system, this will necessitate modifying the system as follows: the new device will have to have a subscriber function block with the same address as the former feeder and an interaction protocol for choosing which feeder to assign the LOAD task will have to be programmed. Programming an interaction protocol can be quite tedious, and doing so in a function block application will add some level of complexity to the system. This is in contrast to the agent-based system where the interaction protocol is already integrated into the software.

6.2.3 Convertibility

This is a measure of how easily conversion can be achieved in the intervals between different production batches. Conversion may involve changing tools, part-programs and fixtures (Koren et al, 1999).

Conversion is possible in both the multi-agent systems and the IEC 61499 control system, and it will take about the same amount of work and time. An example of a possible conversion process could be the following: if the robot gripper is to be changed to accommodate a new type of part to be loaded, the conversion will be restricted only to changes at the level of the robot control program. The changes may include the manual labour of changing the robot gripper, making adjustments for the length of the new gripper in the robot control program, as well as recalibrating the robot. The functions of both the multi-agent and the IEC 61499 control systems will remain the same. If, on the other hand, a different robot is to be used in place of the current robot, the software system remains unaffected if the robot program of the former robot will work for the new. If, on the other hand, the robot programs are different, then a new robot agent or a new robot function block will have to be developed. The time spent in either case of developing a new robot agent or a new robot function block will depend on the programmer. Once the agent or function is completed, the multi-agent system or the IEC 61499 control system is updated and the execution begins.

6.2.4 Diagnosability

This is a measure of the system's ability to quickly identify the sources of quality and reliability problems (ElMaraghy, 2006).

The agent-based control system in this case study is more diagnosable than the IEC 61499 control system. This is because FBDK on which the IEC 61499 control system runs has no debugging tools, while JADE on which the multi-agent system runs uses the debugging tools of Java which hosts the JADE platform. However, FBDK can also be run from a Java environment, for example Eclipse or Netbeans. If this is done, then the debugging tools of Java become available to be used with FBDK and in that case, the diagnosability of the agent-based and the IEC 61499 control systems will be the same.

6.2.5 Customization

This has two aspects, according to Koren et al (1999), which are customized flexibility (i.e. flexibility that revolved around specific parts currently being manufactured only) and customized control (i.e. integration of control modules with the aid of open-architecture technology).

The agent-based control system and the IEC 61499 control system have the same level of customized flexibility, but the IEC 61499 control system has higher customized control than the agent-based control system. The first aspect of customization called customized flexibility is machine dependent according to Koren et al (1999), and therefore, it does not really affect the control systems. On the other hand, the second aspect called customized control has to do with the use of open-architecture technology, particularly for communication. The idea behind the IEC 61499 based control is that the control system should be able to use any hardware or communication protocol independent of the manufacturer. For this to be achieved, the IEC 61499 standard specifies that hardware manufacturers must provide service interface function blocks that can be used to communicate with or access the functionality of their hardware or communication protocol. Once these function blocks are provided, the IEC 61499 control system is able to use the hardware or communication protocol, and control can be customized for that hardware or communication protocol. For agent-based control, the multi-agent systems are, by default, restricted to Ethernet communication. This is because agents typically run on PCs, and communication between PCs is by use of Ethernet. Therefore the agents are restricted to communicate over Ethernet.

6.2.6 Scalability

This is a measure of how the system's capacity can be incrementally changed rapidly and economically (ElMaraghy, 2006).

The agent-based control system is more scalable than the IEC 61499 control system. If the change in system capacity is to be done by changing the capacity of the individual machines, then the control systems will remain unaffected provided only the machine hardware changes. Examples of this include changing the camera's lens or the size of the feeder's hopper, etc. On the other hand, if changing the system's capacity is to be done by adding new machines, for example by including additional feeders, cameras, or robots in the system, then the case is similar to that of integrability. From the integrability of both control systems, it is evident that capacity change will occur much more rapidly in the multi-agent system than in the IEC 61499 control system.

6.2.7 Fault tolerance

The agent-based control system is expected to be more fault tolerant than the IEC 61499 distributed control system. This is because the ability to reroute processes when a subsystem fails is inherent to the agent-based control method, while only scenarios that were foreseen ab initio, can be provided for in an IEC 61499 distributed control application.

In summary, for the case study considered, the agent-based control system is more integrable, diagnosable, scalable and fault tolerant than the IEC 61499 distributed control system. On the other hand, the IEC 61499 distributed control system is more modular and customizable.

7 Conclusions and recommendations

The thesis considers reconfigurable control strategies in a welding assembly cell. Two systems in the assembly cell were considered: the pallet magazine and the feeding system. The pallet magazine control program was written using a PC and a PLC as controllers, while the reconfigurable control of the feeding system was done using the agent-based control method and the distributed control methodology based on IEC 61499. The development of the control for these two systems led to two further objectives:

- the comparison between the PC and the PLC used to control the pallet magazine
- the comparison between the agent-based control methodology and the distributed control methodology based on IEC 61499, which were used in the control of the feeding system.

From this work, it can be concluded that in a reconfigurable system, the hardware and software components of the system have to be modular. There must be proportional modularity between the hardware and the software components in order to ease reconfiguration. For example, the feeding system in our case study can easily be reconfigured since the hardware components (i.e. the singulation unit, the camera and the robot) are separate and their control software are also separate. On the other hand, the pallet magazine cannot easily be reconfigured without having to change its control software, e.g. the control software must change if the drive inverter for the magazine assembly motor changes. This is corroborated by Bi et al (2007b) who state that "system reconfigurability can be classified in terms of the levels where the reconfigurable actions [occur] … the reconfigurability at lower levels is mainly achieved by changing hardware resources, and the reconfigurability at higher levels is mainly achieved by changing software resources. However, they usually work together so that system reconfiguribility can be maximized cost efficiently."

Based on the comparison between the PC and the PLC, it was observed that the PC is more adaptable than the PLC. However, the PLC has a higher capability, availability and usability than the PC. For most industrial applications, the most important attributes of the controllers used are the capability and reliability of the controllers. Concerning these two attributes the PLC fared better than the PC. This, perhaps, explains the preponderance of the use of PLCs in industrial control applications. However, the PC surpasses the PLC with regards to some other attributes (e.g. adaptability), and as Bruccoleri (2005) states: "PC-based control environments bring indisputable advantages [with] respect to PLC-environments especially concerning their networking predisposition, real-time monitoring and visualization capabilities, and flexibility related to the great number of programming languages that could implement the real control software", therefore, a combination of the qualities of PCs and PLCs should be considered. The superiority of the PC over the PLC in terms of adaptability recommends the use of PCs in reconfigurable systems or at least as an augmentation to PLCs. Some manufacturers are already attempting to combine the functionality of PCs and PLCs. An example is the new Programmable Automation Controllers (PAC) by the Opto 22 Company.

The comparison between the agent-based control and the IEC 61499 distributed control showed IEC 61499 distributed control systems are more modular and customizable than agent-based control systems. In addition, the events and data in IEC 61499 control systems are decoupled, thereby making them more flexible than agent-based systems (Fletcher, 2001). However, agent-based control systems are more integrable, more diagnosable and more scalable than IEC 61499 distributed control systems. The low integrability and scalability of the IEC 61499 distributed control systems probably explain why Zoitl et al (2007) state that "a full support for dynamic reconfiguration is beyond the scope of the standard [IEC 61499]". In addition, there is the possibility of losses of events in the use of the IEC 61499 control method since there are no "input event and data queues" as stated by Lewis (2001), and this is because the IEC 61499 standard does not make provision for event storage.

From what has been encountered in this work, the following recommendations are made:

- An ontology should be created specifically for manufacturing to include all the processes within manufacturing. In multi-agent systems developed for manufacturing, this will make actions less ambiguous and better defined.
- Research should be conducted into ways of combining the agent-based and the IEC 61499 based approaches to reconfigurable control. One of the effects would be that IEC 61499 may be more integrable and more scalable.
- More research work should be done on the implementation of the IEC 61499 standard because its philosophy fits perfectly well with the requirements for easy reconfiguration.

- Research should be conducted into ways of developing cheaper hybrid controllers by combining the functionality of PCs and PLCs. This should make the hybrid controllers more affordable and widely available.

- An XML standard format could be developed for information and data exchange for machine controllers as XML is being currently integrated with OPC (OLE for process control).

References

Al-Safi Y and Vaytkin V, 2007, ‘An Ontology-Based Reconfiguration Agent for Intelligent Mechatronic Systems’, Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 114 – 126

Bellifemine F, Caire G and Greenwood D, 2004, 'Developing Multi-agent Systems Using JADE', Wiley Series in Agent Technology, 1st Edition, New York: Wiley

Bi Z M, Lang S Y T, Verner M and Orban P, 2007a, 'Development of Reconfigurable Machines', International Journal of Advanced Manufacturing Technology, Vol. 39: 1227 – 1251

Bi Z M, Wang L and Lang S Y T, 2007b, ‘Current Status of Reconfigurable Assembly Systems’, International Journal of Manufacturing Research IJMR, Inderscience Vol. 2 No. 3: 303 – 328

Bi Z M, Lang S Y T, Shen W and Wang L, 2008, ‘Reconfigurable Manufacturing Systems: The State of the Art’, International Journal of Production Research, Vol. 46, No. 4: 967 – 992

Black G and Vyatkin V, 2009, ‘Intelligent Component-Based Automation of Baggage Handling Systems with IEC 61499’, IEEE Transactions on Automation Science and Engineering, Vol. 7, No. 2: 337 – 351

Bruccoleri M, 2005, 'Reconfigurable Control of Robotized Manufacturing Cells', Robotics and Computer-Integrated Manufacturing, Vol. 23: 94 – 106

Burger J N, 2009, 'Reconfigurable Conveyor System and Pallet Magazine', BEng Final Year Project, Department of Mechanical and Mechatronic Engineering, University of Stellenbosch

Bussman S and Sieverding J, 2001, 'Holonic Control of an Engine Assembly Plant: An Industrial Evaluation', Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference, Tucson, Arizona, USA, 2001: 169 – 174

Bussman S, Jennings N R and Wooldridge M, 2004, Multiagent Systems for Manufacturing Control, Berlin: Springer

Bussmann S and Jennings N R, 2003, 'Agent-Based Control Systems', IEEE Control Systems Magazine, June 2003: 61 – 73

Candido G and Barata J, 2007, 'A Multiagent Control System for Shop Floor Assembly', Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 293 – 302

Chirn J and Farlane D, 2000a, 'A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture', Proceedings of Industrial Applications of Holonic and Multi-agent Systems, HolonMas00, London, UK, September

Chirn J and Farlane D, 2000b, 'Application of the Holonic Component-Based Approach to the Control of a Robot Assembly Cell', Proceedings of IEEE Conference on Robotics and Automation, San Francisco, USA, April

Colombo A W, Schoop R and Neubert R, 2006, 'An Agent-Based Intelligent Control Platform for Industrial Holonic Manufacturing Systems', IEEE Transactions on Industrial Electronics, Vol. 53, No. 1: 322 – 337

De Toni A and Tonchia S, 1998, 'Manufacturing Flexibility: A Literature Review', International Journal of Production Research, Vol. 36, No. 6: 1587 – 1617

ElMaraghy H, 2006, 'Flexible and Reconfigurable Manufacturing System Paradigms', International Journal of Flexible Manufacturing System, Vol. 17: 261 – 276

Ermolayev V and Matzke W, 2007, ‘Performance in Industrial Holonic Systems’, Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 383 – 386

Farlane D, Kollingbaum M, Matson J and Valckenaars P, 2001, 'Development of Algorithms for Agent-Based Control of Manufacturing Flow Shops', IEEE International Conference on Systems, Man and Cybernetics, Tucson, Arizona, USA, October

Ferrolho A and Crisostomo M, 2007, 'Intelligent Control and Integration Software for Flexible Manufacturing Cells', IEEE Transactions on Industrial Informatics, Vol. 3, No. 1: 3 – 11

FIPA (Foundation for Intelligent Physical Agents), 2010, [Online]. Available: http://www.fipa.org

Fletcher M, 2001, 'Building Holonic Control Systems With Function Blocks', Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems, IEEE Computer Society, Vol. 1: 247 – 250

Gouyon D, Petin J and Morel G, 2007, 'A Product-Driven Reconfigurable Control for Shop Floor Systems', Studies in Informatics and Control, Vol. 16

Guessoum Z, 2004, 'Adaptive Agents and Multiagent Systems', IEEE Computer Society, Vol. 5, No. 7

JADE (Java Agent Development Framework), 2010, [Online]. Available: http://jade.tilab.com

Jennings N R and Wooldridge M J, 1998, 'Applications of Intelligent Agents', Agent Technology: Foundations, Applications and Markets, New York: Springer-Verlag

Jennings N R, 1999, 'On Agent Based Software Engineering', Journal of Artificial Intelligence, Vol. 117: 277 – 296

Kamioka K, Kamioka E and Yamada S, 2007, 'An RFID Driven Holonic Control Scheme for Production Control Systems', IEEE Computer Society 2007 International Conference on Intelligent Pervasive Computing, Jeju: 509 – 514

Koren Y, Heisel U, Jovane F, Moriwoki T, Pritschow G, Ulsoy A G and Van Brussel H, 1999, 'Reconfigurable Manufacturing Systems', Annals of the CIRP, Vol. 48, No. 2: 1 – 13

Kotak D, Wu S, Fleetwood M and Tamoto H, 2003, 'Agent-Based Holonic Design and Operations Environment for Distributed Manufacturing', Computers in Industry, Vol. 52: 95–108

Kuo C H, Huang H P, Wei K C and Tang S H, 1999, 'System Modeling and Real-Time Simulator for Highly Model-Mixed Assembly Systems', Transactions of ASME, Journal of Manufacturing Systems and Automation, Vol. 121: 282 – 289

Langer G, 1999, 'HoMuCS – A Methodology and Architecture for Holonic Multi-cell Control Systems', PhD Thesis, Department of Manufacturing Engineering, Technical University of Denmark

Leitao P and Restivo F J, 2006, 'ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control', Computers for Industry, Vol. 57, No. 2: 121–130

Leitao P and Restivo F J, 2008, 'Implementation of a Holonic Control System in a Flexible Manufacturing System', IEEE Transactions on Systems, Man, and Cybernetics, Vol. 38, No. 5: 699 – 709

Lewis R W, 1998, Programming Industrial Control Systems Using IEC 1131-3, London: Institute of Engineering and Technology

Lewis R W, 2001, Modeling Control Systems Using IEC 61499, London: Institute of Engineering and Technology

Liu S, Gruver W A, Kotak D and Bardi S, 2000, 'Holonic Manufacturing System for Distributed Control of Automated Guided Vehicles', IEEE International Conference on Systems, Man, and Cybernetics, Vol. 3: 1727 – 1732

Lohse N, Hirani H and Ratchev S, 2005, ‘Equipment Ontology for Modular Reconfigurable Assembly Systems’, Proceedings of the 3rd International CIRP Conference on Reconfigurable Manufacturing, Ann Arbor, USA: 301 – 314

Mahr T and de Weerdt M, 2007, ‘Auctions with Arbitrary Deals’, Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 37 – 46

Martinsen K, Haga E, Dransfeld S, and Watterwald L E, 2007, 'Robust, Flexible and Fast Reconfigurable Assembly System for Automotive Air-brake Couplings', Intelligent Computation in Manufacturing Engineering, Vol. 6

McFarlane D C and Bussmann S, 2000, 'Developments in Holonic Production Planning and Control', International Journal of Production Planning and Control, Vol. 11, No. 6: 522 - 536

Mehrabi M G, Ulsoy A G, Koren Y and Heytler P, 2002, 'Trends and Perspectives in Flexible and Reconfigurable Manufacturing Systems', Journal of Intelligent Manufacturing, Vol. 13: 135 – 146

Meng F, Tan D and Wang Y, 2006, 'Development of Agent for Reconfigurable Assembly System with JADE', Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China, 21 – 23 June

Monostori L, Vancza J and Kumara S R T, 2006, 'Agent-Based Systems for Manufacturing', Annals of the CIRP, Vol. 55 No. 2: 697 – 720

Movidrive, 2006, 'Modulo Positioning Application', SEW EURODRIVE, Bruchsal

Onofrio C and Bruccoleri M, 2006, 'A User-friendly Control System to Easy Reconfigure a Manufacturing Cell', International Conference on Industrial Technology, ICIT 2006, Mumbai, 15 – 17 December

Padgham L and Winikoff M, 2004, Developing Intelligent Agent Systems: A Practical Guide, Sussex: John Wiley and Sons

Partial-order Verification Techniques, 2004, [Online]. Available: http://www.fmi.uni-stuttgart.de/szs/research/resources/pom/

Paolucci M and Sacile R, 2005, Agent-Based Manufacturing and Control Systems, London: CRC Press

Qiu R G, Joshi S and Mcdonnell P, 2004, 'An Approach To Regulating Machine Sharing In Reconfigurable Back-End Semiconductor Manufacturing', Journal of Intelligent Manufacturing, Vol. 15: 579 – 591

Raj T, Shankar R and Suhaib M, 2007, 'A Review of Some Issues and Identification of Some Barriers in the Implementation of FMS', International Journal of Flexible Manufacturing System, Vol. 19: 1 – 40

Rooker M N, Sunder C, Strasser T, Zoitl A, Hummer O and Ebenhofer G, 2007, 'Zero Downtime Reconfiguration of Distributed Automation Systems', Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 326 – 337

Rzevski G, Skobelev P and Andreev V, 2007, 'Magenta Toolkit: A Set of Multi-agent Tools for Developing Adaptive Real-Time Applications', Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 303 – 313

Scholz-Reiter B and Freitag M, 2007, 'Autonomous Processes in Assembly Systems', Annals of the CIRP, Vol. 56: 712 – 730

Sequeira M A, 2008, 'Conceptual Design of a Fixture-Based Reconfigurable Spot Welding System', MScEng Thesis, Department of Mechanical and Mechatronic Engineering, University of Stellenbosch

Setchi R M, and Lagos N, 2004, 'Reconfigurability and Reconfigurable Manufacturing Systems: State-of-the-Art Review', IEEE Conference on Industrial Informatics, INDIN 2004, Berlin, 24 – 27 June

Shen W, Maturana F and Norrie D H, 2000, 'MetaMorph II: An Agent-Based Architecture for Distributed Intelligent Design and Manufacturing', Journal of Intelligent Manufacturing, Vol. 11: 237 – 251

Strauss R, 2009, 'Development of a Reconfigurable Parts Feeder for Assembly Automation', BEng Final Year Project, Department of Mechanical and Mechatronic Engineering, University of Stellenbosch

Sugi M, Maeda Y, Aiyama Y and Arai Y, 2003, 'A Holonic Architecture for Easy Reconfiguration of Robotic Assembly Systems', IEEE Transactions on Robotics and Automation, Vol. 19, No. 3: 457-464

Tang H P and Wong T N, 2005, 'Reactive Multi-agent System For Assembly Cell Control', Robotics and Computer-Integrated Manufacturing, Vol. 21: 87 – 98

Thramboulidis K, Doukas G, Frantzis A, 2004, 'Towards An Implementation Model For FB-Based Reconfigurable Distributed Conrol Applications', Proceedings of the 7th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing: 193 – 200

TORERO, [s.a.], [Online]. Available: http://www.uni-magdeburg.de/iaf/cvs/torero

Tozicka J, Pechoucek M, Rehak M and Prokopova M, 2007, 'Multi-agent Reflection in Autonomic Systems', Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 27 – 36

Turgay S, 2008, 'Agent-Based Flexible Manufacturing System (FMS) control', Robotics and Computer Integrated Manufacturing, Vol. 25: 470 – 480

Ulieru M, 1997, 'Soft Computing Issues in the Intelligent Control of Holonic Manufacturing Systems', North American Conference on Fuzzy Information Processing Systems, Syracuse, New York, 21 – 24 September

UMI, 1986, 'Introducing RTX', Universal Machine Intelligence Limited, London

Van Brussel H, Wyns J, Valckenaers P, Bongaerts L and Peeters P, 1998, 'Reference Architecture For Holonic Manufacturing Systems: PROSA', Computers in Industry, Vol. 37: 255 – 274

Vyatkin V, 2006, 'The Potential Impact of the IEC 61499 Standard on the Progress of Distributed Intelligent Automation', International Journal of Manufacturing Technology and Management, Vol. 8: 1 – 3

Vyatkin V, 2007, 'IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design', North Carolina: Instrumentation, Systems and Automation Society, ISA

W3C, 2010, [Online]. Available: http://www.w3.org

Wane, S, [s.a.], [Online]. Available: http://www.staffs.ac.uk/personal/engineering and technology/sow1/Robotics/RTX/rtx.htm

Wang F, Guang H X and Min T, 2005, 'An Agent-Based Holonic Architecture For Reconfigurable Manufacturing Systems', Lecture Notes in Computer Science, Vol. 3612, Part III: 622 – 627

Wiendahl H P, 2007, 'Changeable Manufacturing – Classification, Design and Operation', Annals of CIRP, Vol. 56: 783 – 809

Williams D J, 1991, Manufacturing cells: Control, Programming and Integration, London: Butterworth-Heinemann

Yu J, Yin Y and Zhaoneng C, 2005, 'Control Model For Reconfigurable Assembly Systems', High Technology Letters, Vol. 11, No. 2: 171 – 174

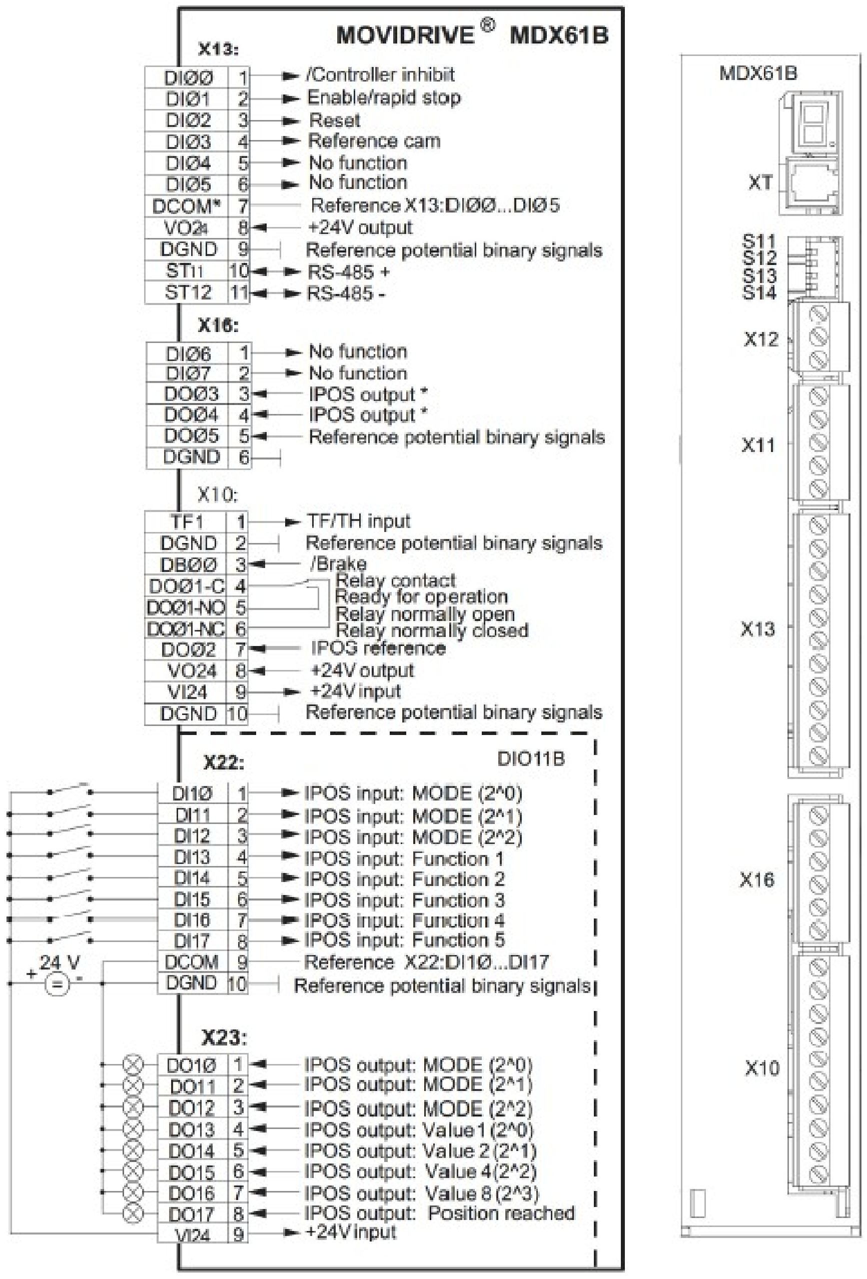
Yu J, Yin Y, Sheng X and Chen Z, 2003, 'Modelling Strategies for Reconfigurable Assembly Systems', Journal of Assembly Automation, Vol. 23, No. 3: 266 – 272

Zhao F, Zhang Q and Wang L, 2008, 'Task Allocation and Evaluation Model For Holonic Manufacturing System Based on Multi-agent System', Proceedings of the 7th World Congress on Intelligent Control and Automation, Chongqing, China, 25 – 27 June

Zoitl A, Strasser T, Hall K, Staron R, Sunder C and Favre-Bulle B, 2007, 'The Past, Present and Future of IEC 61499', Proceeding of the 3rd International Conference on Industrial Applications of Holonic and Multi-agent Systems, HOLOMAS 2007, Regensburg, Germany: 1 – 14

Appendix A Drive inverter control

A.1 Wiring diagram for MOVIDRIVE MDX61B



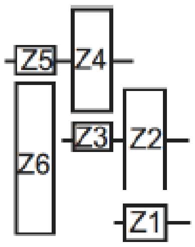
MOVIDRIVE® MDX61B
X13:
DIØØ 1 /Controller inhibit
DIØ1 2 Enable/rapid stop
DIØ2 3 Reset
DIØ3 4 Reference cam
DIØ4 5 No function
DIØ5 6 No function
DCOM* 7 Reference X13:DIØØ...DIØ5
VO24 8 +24V output
DGND 9 Reference potential binary signals
ST11 10 RS-485 +
ST12 11 RS-485 -
X16:
DIØ6 1 No function
DIØ7 2 No function
DOØ3 3 IPOS output *
DOØ4 4 IPOS output *
DOØ5 5 Reference potential binary signals
DGND 6
X10:
TF1 1 TF/TH input
DGND 2 Reference potential binary signals
DBØØ 3 /Brake
DOØ1-C 4 Relay contact
DOØ1-NO 5 Ready for operation
DOØ1-NC 6 Relay normally open
Relay normally closed
DOØ2 7 IPOS reference
VO24 8 +24V output
VI24 9 +24V input
DGND 10 Reference potential binary signals
X22:
DIO11B
DI1Ø 1 IPOS input: MODE (2^0)
DI11 2 IPOS input: MODE (2^1)
DI12 3 IPOS input: MODE (2^2)
DI13 4 IPOS input: Function 1
DI14 5 IPOS input: Function 2
DI15 6 IPOS input: Function 3
DI16 7 IPOS input: Function 4
DI17 8 IPOS input: Function 5
24 V
+
-
DCOM 9 Reference X22:DI1Ø...DI17
DGND 10 Reference potential binary signals
X23:
DO1Ø 1 IPOS output: MODE (2^0)
DO11 2 IPOS output: MODE (2^1)
DO12 3 IPOS output: MODE (2^2)
DO13 4 IPOS output: Value 1 (2^0)
DO14 5 IPOS output: Value 2 (2^1)
DO15 6 IPOS output: Value 4 (2^2)
DO16 7 IPOS output: Value 8 (2^3)
DO17 8 IPOS output: Position reached
VI24 9 +24V input
MDX61B
XT
S11
S12
S13
S14
X12
X11
X13
X16
X10

Figure A.1 MOVIDRIVE wiring diagram (Movidrive, 2006)

A.2 Determining Modulo parameters

The followings steps are used to deduce the Modulo parameters for Modulo positioning.

Motor Gear Box



The values of the gear teeth are as follows – Pinion Z1 - 23T, Gear Z2 - 31T, Second gear pinion shaft Z3 - 18T, Gearwheel Z4 - 97T, Rose Z5 - 8T and Bevel Z6 - 33T.

Gear box numerator = Z2 x Z4 x Z6 = 31 x 97 x 33 = 99231

Gear box denominator = Z1 x Z3 x Z5 = 23 x 18 x 8 = 3312

Additional Gear

Gear ratio is 5:2

Modulo Parameters

Modulo parameters is product of gear box ratio and additional gear ratio.

Modulo parameters = $\frac{99231}{3312} \times \frac{5}{2} = \frac{33077}{1104}$

Therefore, Modulo numerator is 33077 while Modulo denominator is 1104.

Appendix B Vision sensor scripts

Background and Foreground Scripts

Foreground script

```
class partScript
{
        public void inspect()
        {
                //set datalink output inactive with user output User1
                SetOutputs(0L, (1L<<32)); //User1 is output bit 32
                partScript.Result = -1; //assumes it has failed
                if (identifPart_Tool.Result == 0) //Part was correctly identified
                {
                        if (Pickpos_Tool.Result == 0) //if model 1 is the model identified
                        {
                        //get pick point position
                        partScript.Point.X = Pickpos_Tool.PickPoint.X;
                        partScript.Point.Y = Pickpos_Tool.PickPoint.Y;
                        partScript.ANGLE = Pickpos_Tool.PickPoint.Angle;
                        }
                        if (Pickpos1_Tool.Result == 0) //if model 2 is the model identified
                        {
                        //get pick point position
                        partScript.Point.X = Pickpos1_Tool.PickPoint.X;
                        partScript.Point.Y = Pickpos1_Tool.PickPoint.Y;
                        partScript.ANGLE = Pickpos1_Tool.PickPoint.Angle;
                        }
                }
                //set the external trigger mode bit to 1 (i.e. on) to stop inspection
                SetInputs((1L<<7), 0L);
                //set datalink output active with user output User1
                SetOutputs((1L<<32),0L ); //User1 is output bit 32
                partScript.Result = 0; //shows it is completed
        }
}
```

Background Script

```
class NewScript
{
        public static void main(String args[])
        {
                while(true)
                {
                        int len = 2; //This is the length of data to be received
                        int port = 3248; //The port no where the camera listens for connection
                        int conStatus = -1; //Detects when a connection is accepted
                        byte data[] = new byte[len]; //Byte array to receive the incoming data
                        Socket mySocket = new Socket(); //New socket object
                        Socket sock = new Socket(); //socket generated to accept a connection
                        int status = mySocket.Bind(port);
                        if (status==0)
                        {
                                status = mySocket.Listen();
                                if (status == 0)
                                {
                                        while (conStatus != 0)

                                        {

                                                conStatus = mySocket.Accept(sock);

                                        }

                                        status = sock.Recv(data, 0, len); //Receives data
                                //Check whether system is busy and wait until it is idle

                                        while((GetOutputs() & (1L<<8)) != 0) //Bit 8 (busy bit)

                                        {//Waiting for current inspection to end

                                        }

                                        //Handling the case of inspection command

                                        if (data[0] == 1) //An inspection command

                                        {       //set the trigger bit to 1

                                                SetInputs(1L, 0L);

                                                //set the external trigger mode bit to 0

                                                SetInputs(0L,(1L<<7));
                                                //get product object

                                        Product prod = GetProductById((short) data[1]);

                                                prod.Select(); //sets the inspection product

                                                Inspect(); //begin inspection

                                        }

                                }

                        }

                        // Short delay before next iteration

                        sleep(10);

                }

        }

}
```

End of scripts

Appendix C Controller ports and XML data format

C.1 Port Designation for Controllers

The port assigned to each controller is given after the colon (:).

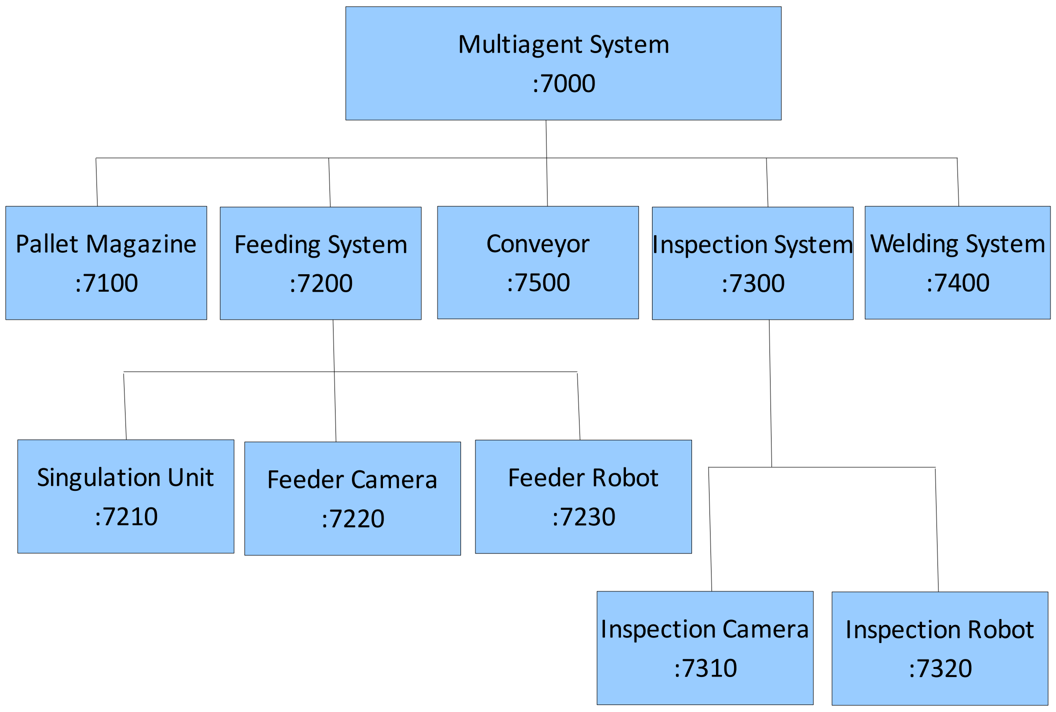


Figure C.1 Controller ports

C.2 XML data format adopted in this work

The format shows the standard request and response formats. In the request format, “sender” is the controller/agent sending the request message (e.g. camera agent), “receiver” represents the intended receiver (e.g. camera controller), “task” is the job to be done (e.g. LOAD) and that is filled in the {…}, while “req” is the data needed to do the job (e.g. part type, etc). Additional data, e.g. X, Y, and Z coordinates of the pick point as for the case of the robot, may be supplied.

The situation is similar for the response format, except that <done> will only have a Boolean value to indicate whether the task was done successfully or not. Additional information may be supplied, e.g. the case of the camera that gives the X, Y and Z coordinates of the pick point.

REQUEST MESSAGE

```
<?xml version="1.0" encoding ="UTF-8"?>

<sender>

        <receiver>

                <task>{...}</task>

                <req>{...}</req>

                …

                ...

        </receiver>

</sender>
```

RESPONSE MESSAGE

```
<?xml version="1.0" encoding ="UTF-8"?>

<sender>

        <receiver>

                <task>{...}</task>

                <done>{...}</done>

                …

                ...

        </receiver>

</sender>
```

Appendix D Feeding system function blocks

IEC 61499 Function Blocks for Feeding System

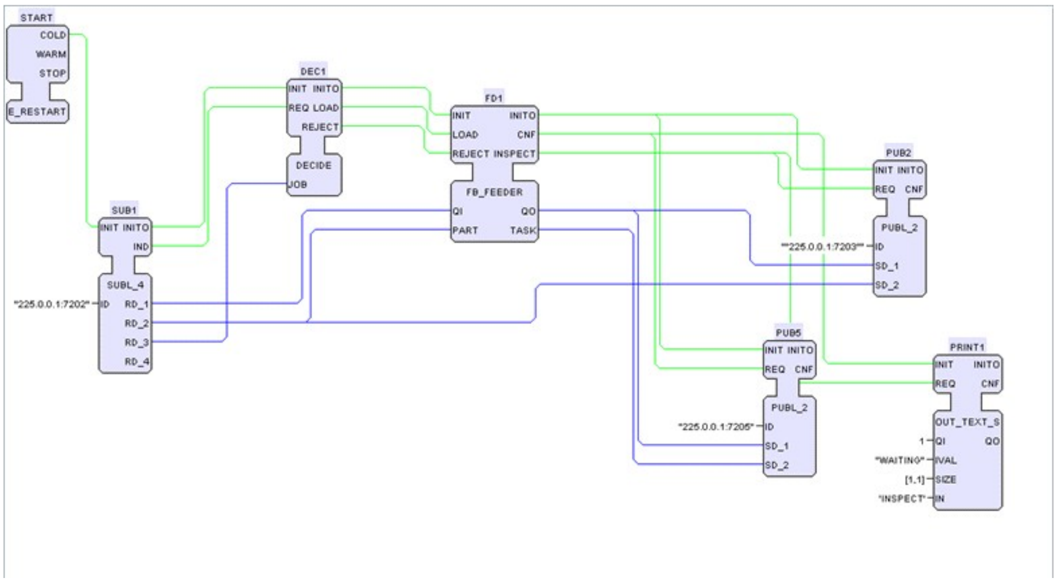


Figure D.1 Feeder device function blocks

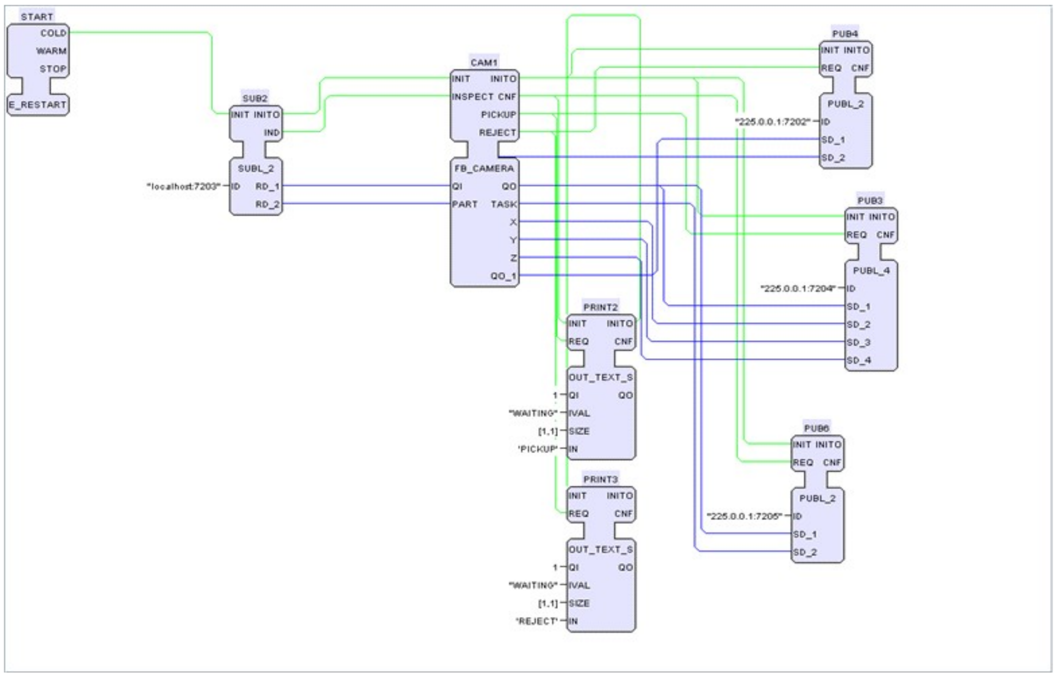


Figure D.2 Camera device function blocks

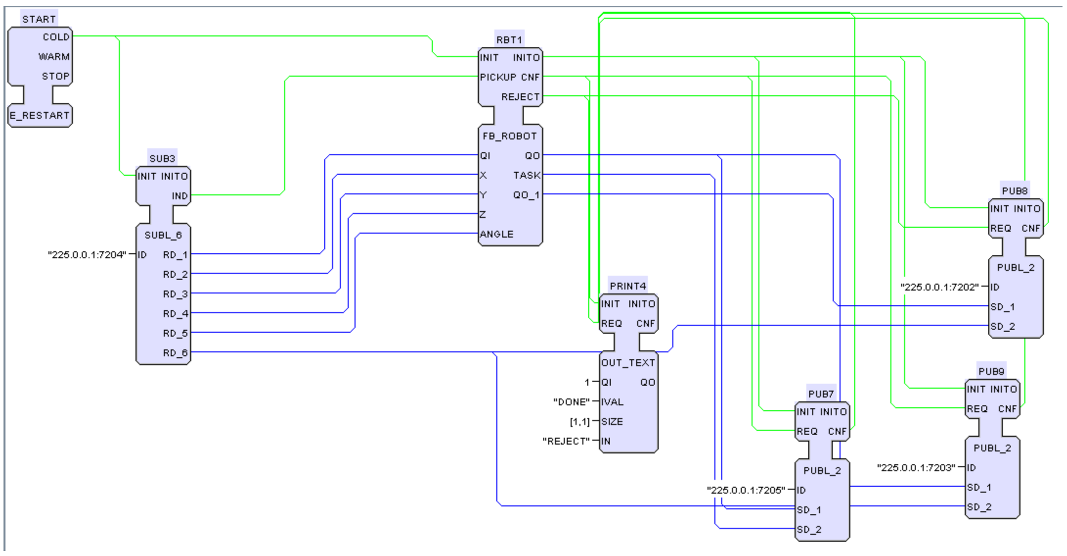


Figure D.3 Robot device function blocks

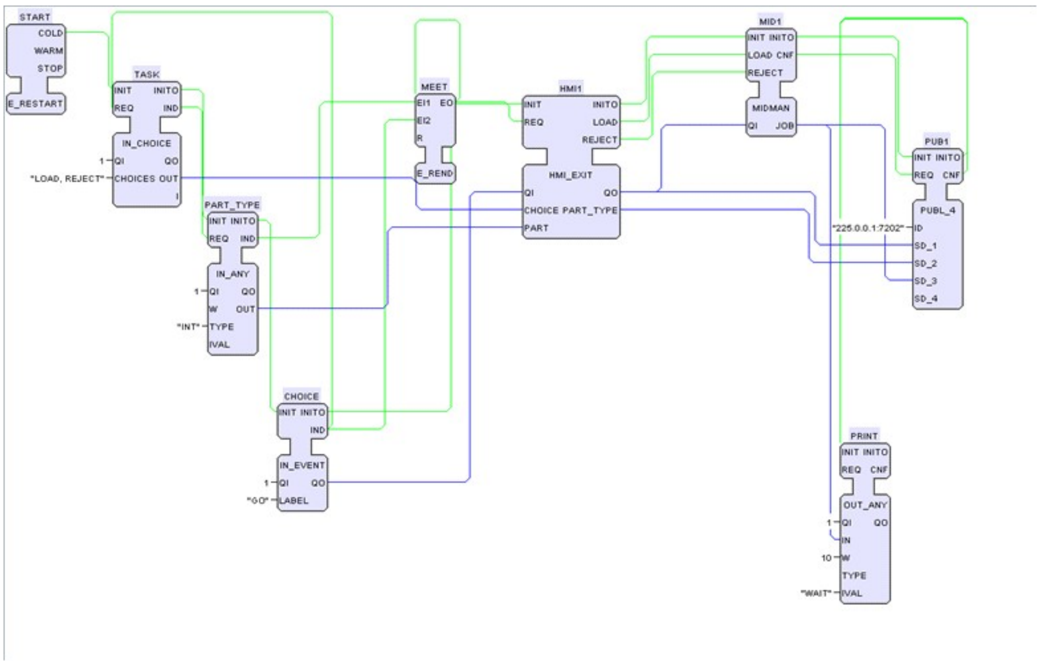


Figure D.4 IN_DEV1 input HMI function blocks

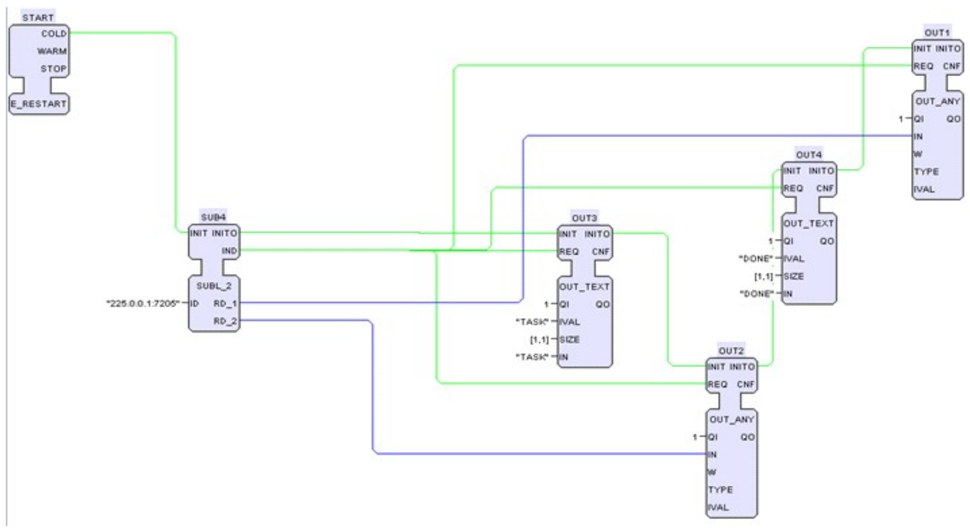


Figure D.5 OUT_DEV1 output HMI function blocks