Design and Development of RFID-enabled Flexible Manufacturing Cell Control System

Majid Mohammad Sadeghi

Submitted to the
Institute of Graduate Studies and Research
in partial fulfilment of the requirements for the Degree of

Master of Science
in
Mechanical Engineering

Eastern Mediterranean University
August 2013
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Mechanical Engineering.

Assoc. Prof. Dr. Uğur Atikol
Chair, Department of Mechanical Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Mechanical Engineering.

Prof. Dr. Majid Hashemipour
Supervisor

Examining Committee

1. Prof. Dr. Majid Hashemipour
2. Asst. Prof. Dr. Ghulam Hussain
3. Asst. Prof. Dr. Neriman Özada

ABSTRACT

Nowadays organizations challenge each other in highly competitive market, in which customer’s demands change frequently and swiftly. In these circumstances, organizations have to be very flexible in respond to customers to be able to survive and prosper. This flexibility lies within the control mechanism in production. Current control systems in production and manufacturing encompass some deficiencies in providing highly flexible systems. In this thesis, RFID technology has been used in manufacturing control system to overcome some of the deficiencies of current control systems. A RFID enabled control system is designed and implemented for a manufacturing cell. The proposed control system has many advantages brought by RFID technology including: bringing the real-time monitoring ability to the system, facilitating quality control and fault finding, reducing errors in management of raw material handling, production operations and finished products, enabling product customization, providing capability of reacting to incidents at production time, creating the chance of real-time customization of products and improvement of just in time manufacturing paradigm and at last providing capability of simultaneous multiple type production with no predefined order. All the proposed beneficial characteristics of the system are tested and verified at implementation phase and it is observed that the control system is absolutely applicable in industry.

Keywords: Manufacturing Control System, Radio Frequency Identification (RFID), Flexible Manufacturing Cell

ÖZ

Bugünkü kuruluşlar, müşterilerin hızla ve sıklıkla değişen ihtiyaçlarından dolayı oluşan yüksek rekabetli piyasada birbirlerine meydan okumaktadırlar. Bu durumda, kuruluşlar hayatta kalmak için müşterilere çok esnek davranmaktadırlar. Bu esneklik üretimdeki kontrol mekanizmalarını da içermektedir. Akım kontrol sistemleri, üretimde ve üretimi kapsayacak bazı eksikliklerde sistemlerdeki yüksek esnekliği sağlamaktadır. Bu tezde kullanılan RFID teknolojisi, üretim kontrol sistemlerinde, akım kontrol sistemlerinin bazı eksikliklerinin üstesinden gelmek için kullanılmaktadır. RFID etkin kontrol sistemleri, üretim hücreleri için kontrol edilmektedir ve uygulanmaktadır. RFID teknoloji tarafından önerilen kontrol sistemleri çok avantaj kazandırmaktadır. RFID teknoloji sisteme izleme yeteneği, kolaylaştırılan kalite kontrol ve arıza tespiti, üretimde bulunan ham materyal tutumunda hataların azalması, üretim operasyonları ve ürün bitimi, etkin ürün özelleştirme, üretim zamanında olaya tepkinin sağlanması, gerçek zamanlı üretimin yaratılma şansı ve sonda önceden tanımlanmamış siparişle eş zamanlı çoklu çeşit üretim yeteneği sağlaması getirmektedir. Bütün önerilen sistemin karakteristiği test edilmiştir ve uygulama aşamamasında onaylanmıştır ve gözlemlenmiştir. Üretimde kontrol sistemi kesinlikle uygulanabilmektedir.

Anahtar Kelimeler: Üretim Kontrol Sistemi, Radyo Frekans Tanıma (RFID), Esnek Üretim Hücresi.

To My Family & My Beloved Wife

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Professor Dr. Majid Hashemipour who has fully supported me through my entire master thesis. Without his comprehensive guide this work would have not been completed.

I also want to thank all the faculty members of Mechanical Engineering Department who have helped me a lot during my master studies.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURES

Chapter 1

INTRODUCTION

Due to the increased pace of changes in the market demands, global competition and also the rate at which technology alters, manufacturers must seek a way to be able to deal with uncertainties. It is believed that flexibility in manufacturing is a proper solution to the mentioned problem (Sethi & Sethi, 1990). The definition of flexibility in general is given by Upton, (1995) as "the ability to change and react with little penalty in time, effort, cost, or performance". Weak reaction to changes will lead to longer production life and delays in answering to customer demands. Traditional manufacturing control systems are not competitive enough to adapt to present circumstances. Developing a manufacturing control system to satisfy flexibility needs, as well as productivity and quality, is the challenge faced. RFID and Auto ID technology has been used in overcoming the challenge. Kohn, Brayman, & Littleton (2005) have proposed architecture to improve productivity and efficiency by implementing RFID data into planning and scheduling of an enterprise. The use of RFID in Ford Company to enable just in time manufacturing is presented by Johnson (2002). Huang, Zhang, & Jiang (2007) have shown the improvement in productivity and quality based on the real-time traceability and visibility RFID provides. Hua, Sun, Liang, & Lei, (2008) have discussed use of RFID real-time information to achieve lean production. Sun (2011) has applied RFID in control of product assembly. Liu, Zhang, Ni, & Tseng (2004) have studied RFID application in customization of parts in mass production.

This work proposes a manufacturing control system based on implementing RFID technology in the manufacturing cell to improve flexibility as well as manufacturing efficiency. First an introductory description of flexibility and flexible manufacturing system is presented. Then control system mechanisms in FMC are introduced along with an introduction to RFID technology. Second chapter analysis current control systems and their deficiencies and what RFID can bring to the control system. In chapter three RFID enabled control system is proposed and described in detail. In chapter four the proposed system implementation and the verification of claimed benefits and functionalities are presented.

1.1 Flexible Manufacturing System

In the field of manufacturing which is the subject of this work, the concept of manufacturing flexibility is investigated. It is defined by ZELENOVIĆ (1982) as the manufacturing system adaptability to environmental conditions changes and changes in the process requirements. Browne, Dubois, Rathmill, Sethi, & Stecke (1984) have classified manufacturing flexibility in eight categories as below:

- Machine flexibility: is referred to how easily changes are applied to produce a new set of part types, or in other words how many operations machine can perform with the same set-up.
- Process flexibility: the number of part types that can be produced in the system without considerable change in the system set-up.
- Product flexibility: the amount of time and cost which is required to change processes to add new part types to existing mix of products.

- Routing flexibility: the number of different available routes to produce a part.
- Volume flexibility: the vastness of range of volumes at which the system can work profitably.
- Expansion flexibility: the ability to expand the system functionality by altering the physical set-up
- Operation flexibility: difference process plans to produce the same part, which means ability to change the order of operations of a part.
- Production flexibility: the total number of part types which can be fabricated in the manufacturing system.

Flexible manufacturing systems and manufacturing cells have been developed based on the need for attaining flexibility and efficiency simultaneously through reducing cost and setup time, improving quality, and boosting reliability of equipment (Chan & Bedworth, 1990).

Flexible manufacturing system is defined as a linked group of machines by a material handling system which is controlled by computer control system and can respond to the changes, which bring flexibility to the system (Shivanand, 2006). A flexible manufacturing system is shown at Figure 1.1.

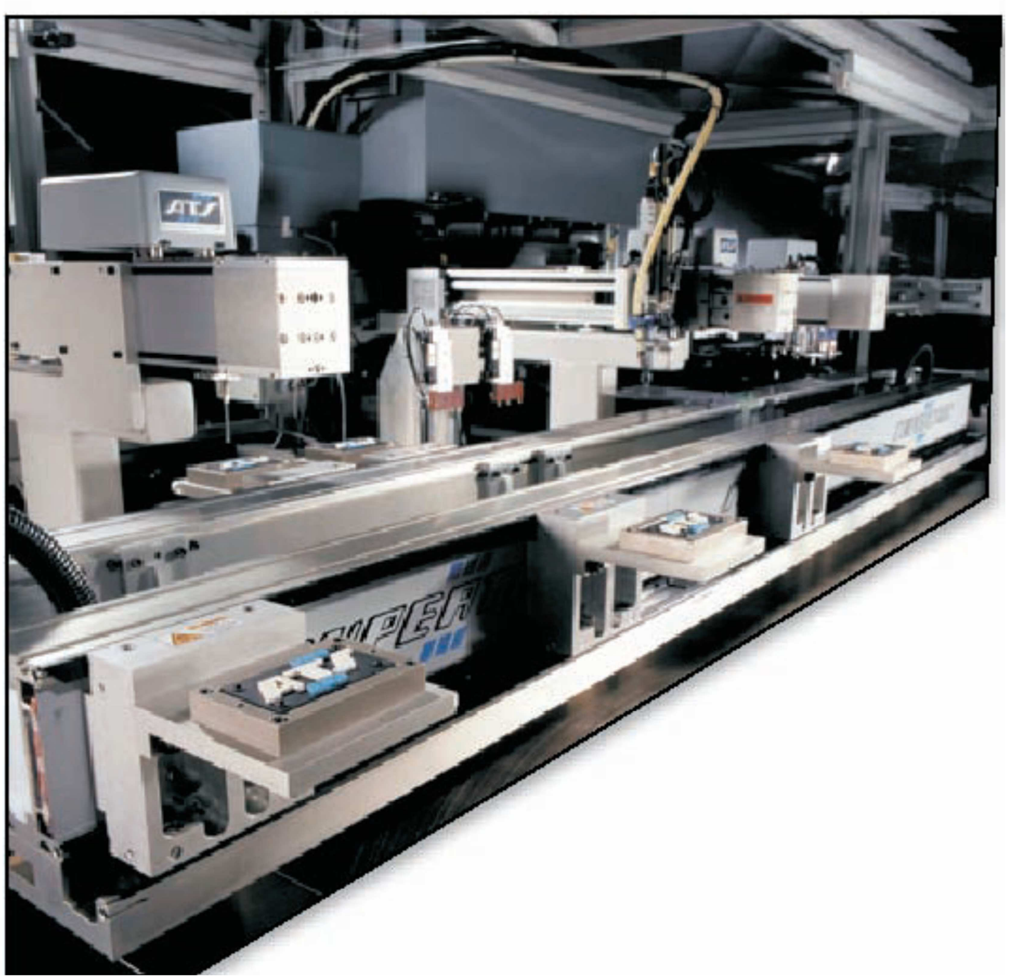


Figure 1.1: Flexible Manufacturing System

Each flexible manufacturing system consists of three main components:

- Workstations, which are usually computer numerical (CNC) machines, but can also include assembly workstations, inspection stations, or any other processing stations.
- Automated material handling system which is responsible for carrying parts between workstations and/or storage and optimize the part flow.
- Computer control system which acts as the controller of each workstation and the material movement and do the monitoring as the basic functions.

Flexible manufacturing systems provide many advantages for the production system based on the amount of each type of flexibility they can possess. These advantages include lowering the cost of production at each part because of higher productivity, more rapid changes in part type production with cheaper cost that leads to more monetary benefits, less inventory costs due to the more precise planning, reducing labor costs as a result of automation and less number of workers needed, improving the quantity of production due to automation and better control system, and omitting a great deal of cost of errors in the production (Chen & Adam Jr, 1991).

1.2 Control Mechanisms in FMS

Control systems in manufacturing are divided into two general categories; low-level control and high-level control. Low-level control deals with how to interact with the manufacturing hardware components such as robot, machines etc. to manipulate them in the desired manner. This type of control is out of the scope of this work. The high-level control is about coordinating activities and resources in manufacturing to reach the aims of the production system in producing parts. High level control systems are categorized into two groups namely centralized and decentralized.

1.2.1 Centralized Control System

This control system is based on a hierarchical architecture that includes variant control units. These control units are organized in a pyramidal design and each level of the system has defined functionalities and objectives. Control units are linked to each other in a way that each unit in upper hierarchy sends control order to the lower units and each lower unit sends the feedback to the upper one.

There are some forms of modified central hierarchical control architectures which are distributed in a way that lower control units can exchange data and strict hierarchy order is replaced by a more coordinating one (Bongaerts, Monostori, McFarlane, & Kádár, 2000).

The centralized control approach offers a good performance which is well predictable and results in a fine production optimization. For the reason that this control system has a high rigidity, it responds weakly to changes. Furthermore, the performance of hierarchical control systems reduces very fast when facing disturbances (Dilts, Boyd, & Whorms, 1991). The reason for that lies in the nature of these control system; at the face of disturbance the feedback is provided to the upper control levels until the problem can be solved and a new schedule can be devised and sent back to the lower levels to react to the disturbance. Thus the robustness and their reaction time are very low. Furthermore the reconfigurability of this control system in case of change in the production type or change based on introducing another manufacturing technology into the system is restricted both from the hardware and software point of view (McFarlane, Sarma, Chirn, Wong, & Ashton, 2003).

1.2.2 Decentralized Control Systems

This type of control system has been developed to overcome the disadvantages of central control systems and follow heterarchical architecture. There is no central control unit, and the control units in this system are greatly distributed. Decisions are made by the interaction of control units which will give this system the advantages of low complexity and better response to changes and disturbances. Architecture of centralized and decentralized control systems are depicted in Figure 1.2.

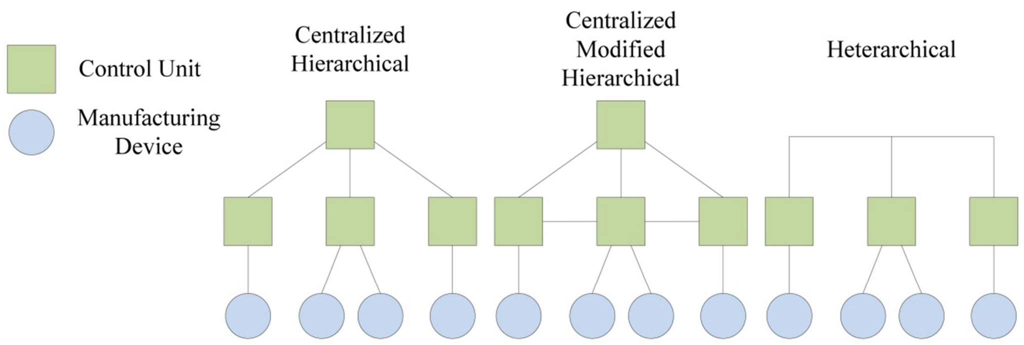


Figure 1.2: Centralized and Decentralized Control Architecture

To achieve the largest amount of machine and resources usage and Swift adjustment to product changes, flexible manufacturing systems' planning and control must act very efficiently. This requires flexibility to be embraced in the control system in a way that it can be adjusted properly and with least trouble at facing disturbances, and this is the fact that moves flexible manufacturing systems towards decentralized control systems (Wong & Leung, 2010).

1.3 Radio Frequency Identification (RFID)

RFID is the leading technology in automatic identification and data collection. This technology makes use of radio frequency in identification, tracing and controlling objects (Hagl & Aslanidis, 2009). When any application needs to communicate with an object(which a tag is attached to it), it sends a command to the middleware and middleware sends a proper command to the RFID reader to send radio wave signals to electronic tag which contains some data such as identification number. Tag sends back a signal to reader which reveals its data and the data is sent back in the same way. By this way the system can identify the presence of the object which the tag is attached to, and the information about the object. An overview of a RFID system is shown in Figure 1.3.

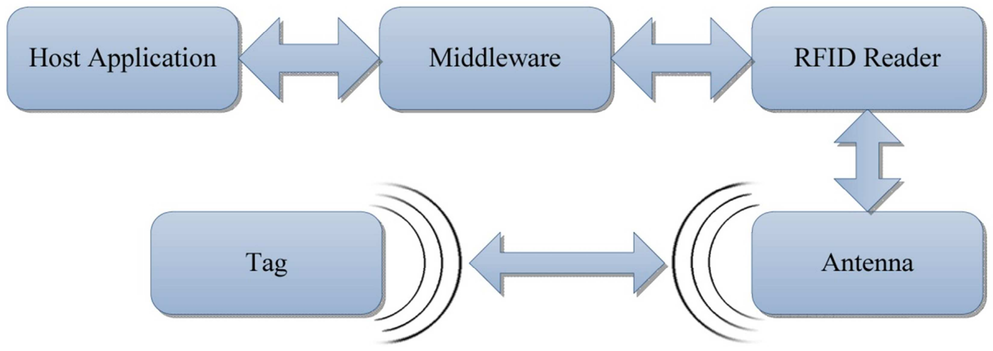


Figure 1.3: Overview of a RFID System

RFID systems consist of hardware components including: tag, antenna, reader and host system, and software components including: RFID system software, RFID middleware and application software.

1.3.1 Hardware Components

1.3.1.1 RFID Tags

Tag functionality is keeping data and transmitting it to the reader. There are three parts inside a tag; an IC (integrated circuit), memory which is optional, and an antenna.

The IC of a tag itself consists of a memory and a microprocessor. The responsibility of microprocessor inside IC is to process the data tag receives from the reader, to make a decision for necessary action. This action can be, for instance, ordering the identification number present in the IC memory to be sent back to the reader. The above communication is feasible through using the tag antenna, which provides and amplifies the radio waves and expand the communication field.

Memory of a passive tag is divided into four parts as shown in Figure 1.4. The reserved tag memory contains passwords which are needed when the reader wants to write data on the tag memory. So the data on this part is used at the time of writing on the tag and if tag does not have a password, this part of memory will have zero value.

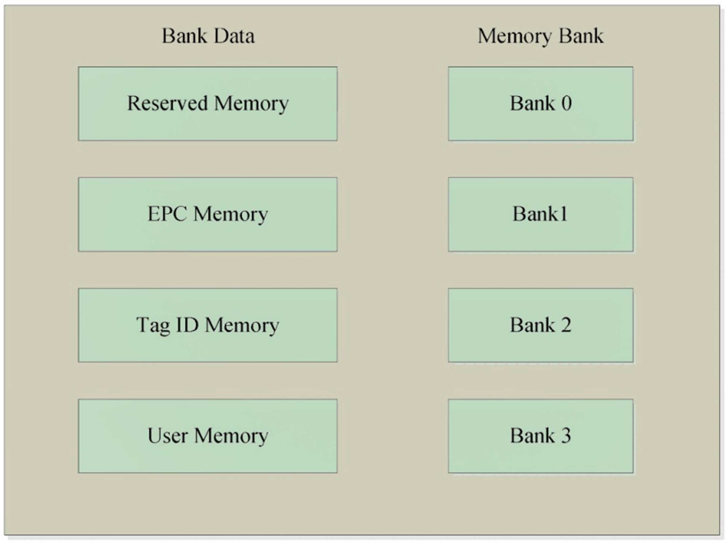


Figure 1.4: Passive Tag Memory Structure

EPC part of the tag memory (Electronic Product Code) is the part which the unique data used to identify the tag is stored at. This is the main part of a passive tag memory. Next is the Tag ID memory part. This part contains information about the manufacturer and type of the tag. The user part of memory of the tag is where data can be written by the user, and depending on the tag type, can have different sizes which give different writing capabilities. The more space on this part, the more data user can save on tag memory (Bolic, Simplot-Ryl, & Stojmenovic, 2010).

Tags are also classified based on their power source to passive and active tags.

1.3.1.1.1 Passive Tags

This type of tag uses the energy of the radio waves it receives, to energize itself and operate the actions, in other words the tag has no internal power source. This gives the tag unlimited life from the power source point of view, since there is no battery implemented in tag to be depleted. The other advantages of passive tags are their cheaper price and their smaller size. The short range of operation of passive tags is their disadvantage in comparison with Active tags.

Passive tags can have battery in some types but the battery is not used to send the signal. The responsibility of battery is to provide enough power for the onboard sensors which are used to sense the environmental conditions like temperature. Some of the passive tags types are shown in Figure 1.5.

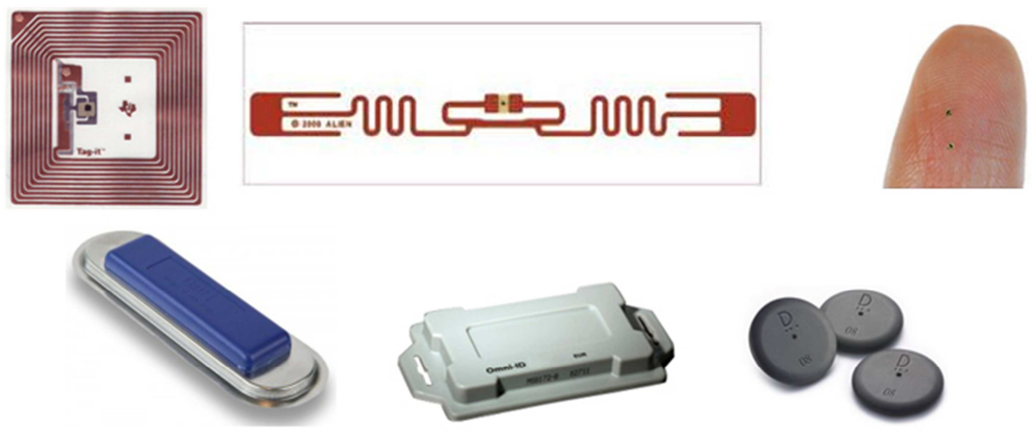


Figure 1.5: Passive RFID Tags

1.3.1.1.2 Active Tags

This type of tags are supplied with internal power source like batteries, which enables it to have longer range of communication due to higher power and a higher signal strength. These tags can be read even when they are not in the reader's field of power. They send signals in a periodic manner and with a constant frequency. So their life span depends on the frequency of signaling which uses the battery. There are drawbacks for active tags including their high cost, and large size, due to the internal power source cost and size.

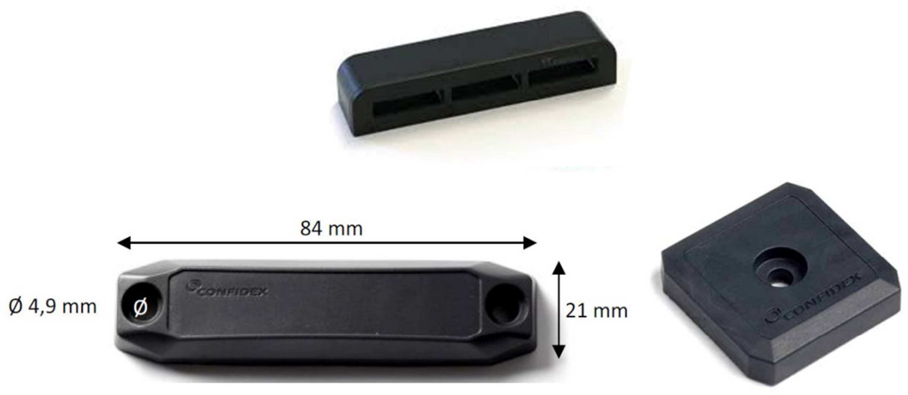
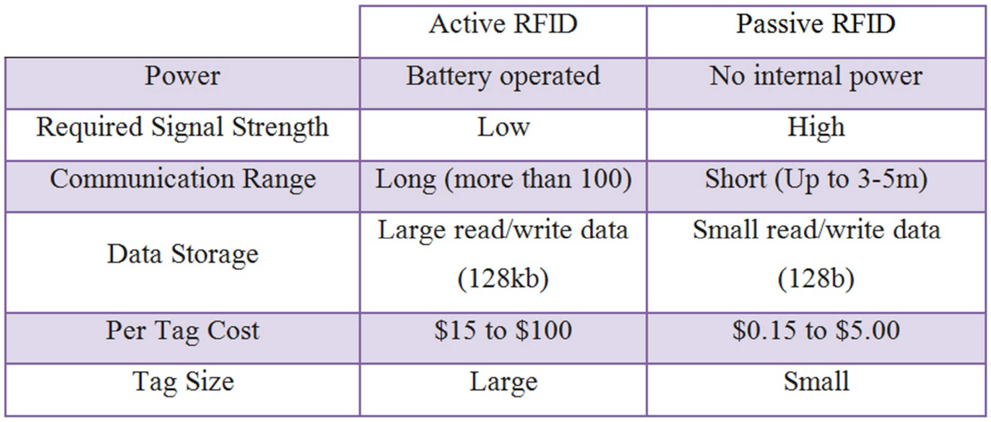


Figure 1.6: Active RFID Tags

1.3.1.1.3 Semi-active Tags

This type of tag contains batteries on board but the battery is in sleep mode until tag receives the signal from the reader power field. At this time the onboard battery activates and tag uses its internal power source to send back the signal. After the operation, the battery gets dormant again which leads to the much longer battery life compared to active tags and much larger communication range compared to passive tags. Semi-active tags are also smaller and cheaper to produce than active tags.

Table 1.1: Active vs. Passive RFIDTags



	Active RFID	Passive RFID
Power	Battery operated	No internal power
Required Signal Strength	Low	High
Communication Range	Long (more than 100)	Short (Up to 3-5m)
Data Storage	Large read/write data (128kb)	Small read/write data (128b)
Per Tag Cost	$15 to $100	$0.15 to $5.00
Tag Size	Large	Small

1.3.1.1.4 Read Only versus Read/Write Tags

Based on the type of memory included in tags they are differentiated into read only or read/write tags. Read only tags have a fixed identity number which can be read by the reader but cannot be changed. This identity number is encoded inside tags in their production time. In addition there are some types of read only tags that can be written only once after the production and during the setup, where they are going to be implemented. This possibility let the user to implement desired coding system for tags.

Read/write tags are capable of being written many times during their lifetime. Their memory contents can be changed and can be used both for identification and also for holding the data related to the state of the tag and other information.

1.3.1.2 RFID Reader and Antenna

Reader is a device that by the help of antenna sends the signal to tags and reads the signals sent by the tags. It provides the host system with the data of the tags. Readers are produced in different types based on their mobility. Handheld readers can be used for mobile jobs whereas fixed readers can be used in the stationary positions.

Readers can be internally powered or use external power source. To communicate with the host system, reader uses a serial or wireless communication.

Readers are composed of two main parts: antenna and IC board. Antenna is the device that converts the electric signal of reader to electromagnetic waves sent to the tag. Different types of antennas are designed for operating different wave frequencies and based on the required size and shape. Some readers even implement antennas inside the reader for the ease of handling. Some types of readers support multiple number antennas. RFID reader antennas are shown in Figure 1.7.

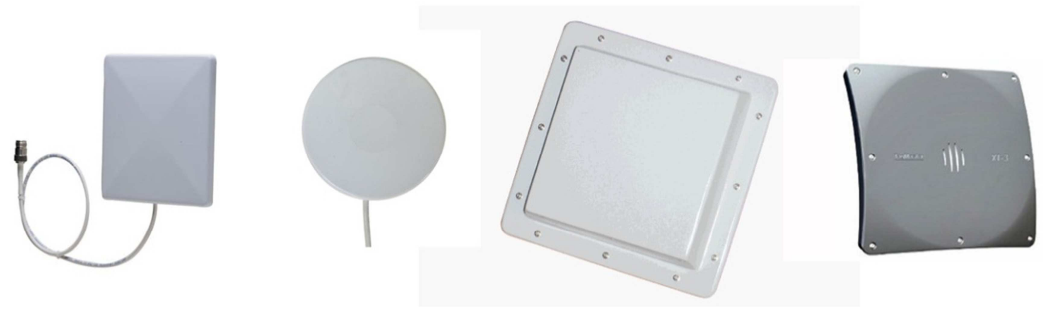


Figure 1.7: RFID Reader Antennas

IC board of the reader arranges the information necessary in communicating with the tag. So it contains a microprocessor to deal with the complex circumstances and communication (Miles, Sarma, & Williams, 2008). For example when reader wants to send a reading signal to the tag, to get the identification number of the tag, the microprocessor handles the signal calculations and sending activities. Some types of RFID readers are shown in Figure1.8.

Figure 1.8: RFID Readers

1.3.2 Software Components

RFID system software components differ to a large extent by the type of system they are integrated in. The first component is RFID System Software, which executes operations between the tag and the reader. These operations between tag and reader demand all hardware, firmware (or low-level software) and high-level system software. System software carry out read and write functions between reader and tags.

The other software components in RFID system is RFID middleware. Middleware is responsible to filter all the tag data coming from the reader and format it in a way that is readable and useful for the application software. It should be considered that there can be a huge amount of tag data during the reading process (Cheong & Lee). Take into consideration that each RFID tag can be read more than 10 times in a second and if there are 100 tags in the reading zone of reader, in a minute there will be information of 60,000 tags in the reader, which is useless without proper filtering.

Host application is other software component of RFID system. This is the application software which makes use of RFID data received from the middleware application to achieve the system goals. For instance inventory control software in an enterprise is the host application which uses the RFID data to track materials and inventory.

1.3.3 Basic Communication Process for Passive Tags

When the enterprise application software sends the read command to the middleware layer of the RFID system, middleware sends this command to the reader at the RFID system software layer, and reader sends an activation signal through antenna to the passive tag. Then tag receives this signal and provides energy for its action from this signal. Next the reader sends the data signal to tag through the antenna which contains the command, the tag carry out this command at this step and then sends the signal back including the results of the comment (Beckner, Simms, & Venkatesh, 2009). The steps of these operations are described in Figure 1.9.

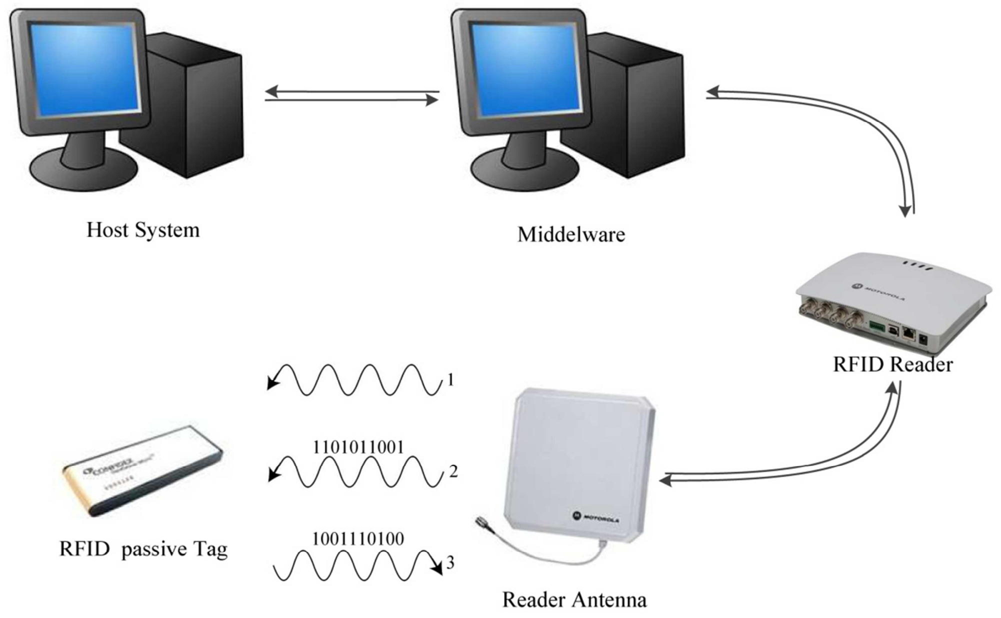


Figure 1.9: Basic Communication Process for Passive Tags

1.3.4 Advantages of RFID System

RFID is used in many different applications including manufacturing, supply chain management, inventory control, access control, automatic payment etc. Characteristics of RFID which have made it the better choice in these applications in comparison to existing identification systems include followings:

- Communication without line of sight requirement
- Large area of communication and operation
- Faster communication speed
- Write ability on memory and high memory capacity
- High security of the information and encryption capacity of data
- Reliability of the system
- Ability of multiple tag reading
- Low power requirements
- No human intervention
- Small size of the tags
- Cost efficiency

Chapter 2

CURRENT CONTROL SYSTEM STUDY

Current manufacturing systems either using centralized or decentralized control systems embody some structural deficiencies. In this chapter those deficiencies and the way RFID can help improve flexible manufacturing is discussed.

First of all in the current manufacturing system, access to activities and the state of the system and parts, and since the real time monitoring capability, are limited in comparison to RFID enabled systems. The information received from RFID is timely and accurate so movement of parts and the system can be monitored in real-time much more accurately.

Secondly, in the production line of current shop floors, parts cannot be differentiated from other similar ones. This problem has blocked many opportunities improving manufacturing systems. Implementing RFID technology to manufacturing system would make each part a unique and identifiable part that can carry its own data in itself thus providing below opportunities:

- Data available on tag attached to parts, recorded during the production, are the source which errors and quality control issues can be tracked from. This option leads to easier fault finding in the system.

- By the help of product information such as identity, specification and states which are available on time by RFID, errors and delays in the management of raw material, production operations and also finished products are reduced (McFarlane et al., 2003).In the flexible manufacturing cell using centralized control system RFID information make it possible to assure correct part arrives to the cell. If parts that are not planned for this manufacturing cell or station are presented at the entry point of the cell they will be removed. In the same way part types with specific order of entry can be sorted before entering to the cell. This function reduces the risk of production errors which insert long delays in the production.
- In the flexible manufacturing systems using decentralized control system customization of products is made much easier with RFID. For instance, when a product which has different options of customization is going to enter the production phase, a RFID tag containing information about the specific details of product is attached to it and during the production it will guide it through the stations which produce those specific features. In this way errors will be eliminated and the process will be much easier and faster.
- Another advantage of RFID in Decentralized control system, where control units are distributed and decisions are made by different units, is the ability of decision-making or reacting to incidents during the production. If part types which are not supposed to be operated in a specific manufacturing cell or workstation accidentally arrive, the irrelevancy can be detected and parts will be rejected to enter to the cell or workstation and can be put out and return to its original root for production. This scenario is also true for finished parts which might enter the production again, and also in assembly

cells. In assembly operations, subassemblies can be detected and chosen properly to go into the relevant workstations. If pertinent parts for this assembly are not arrived at the relevant station, they will not be processed and fault in the assembly operations will be prevented.

Thirdly, concurrent manufacturing approach which tries to reduce the elapsed time to answer to customer orders and needs, can be greatly improved using RFID technology. The real-time customization of parts, or change of production plans for some parts is also feasible when RFID is applied to decentralized control systems. If an order changes during the production based on the needs of customer or technical issues, the new data relating to changed operations can be written on tags and parts will continue to be produced with the new production plan. Hence in the middle of production parts can be customized and there will be an overlap between manufacturing and design process. By creating this overlap, the total time from the customer order to finishing production is shortened.

Finally, in current manufacturing systems, ability of production of various part types at the same time without a specific order is restricted if not possible. By means of RFID in the distributed control, multiple part types can be produced without any predefined order of production at the same time. In the centralized system, operations were decided for a specific number of each part type, and workstations where scheduled to produce each part type with a certain number and a fixed arrangement. However through identifiability of parts with the RFID system, the distributed control system allows different part types to enter the manufacturing cell with no particular order simultaneously, and each work station will detect the part type from the information of the RFID tag attached to it. As a result multiple parts routing will exist in the manufacturing cell.

Chapter 3

RFID ENABLED CONTROL SYSTEM DESIGN

To achieve advantages of RFID technology in the flexible manufacturing system and prove the possibility of benefits argued earlier in the system section, a control system in a flexible manufacturing cell is proposed. The proposed architecture and its functionality are demonstrated in the following sections. The new architecture of RFID enabled flexible manufacturing cell is shown in Figure 3.1.

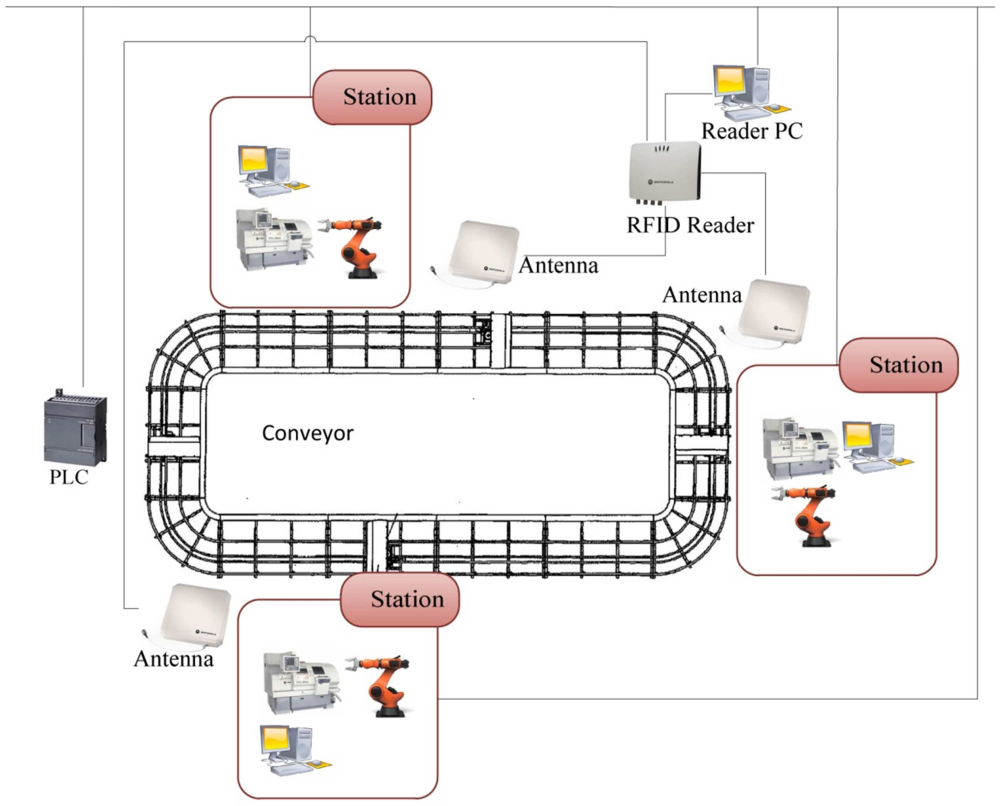


Figure 3.1: Feasible RFID Enabled Manufacturing Cell Layout

There are stations in the cell, each composed of CNC machines, industrial robots and PCs. Stations are placed next to the conveyor. RFID reader is installed in the cell and is connected to a PC. A RFID antenna is placed into each station’s entrance. The conveyor is connected to PLC which itself is connected to the PC through an interface. All PCs are connected together on a network and can exchange information.

Hardware and Software components of the proposed system are described in detail in coming pages.

3.1 Hardware Components

3.1.1 Computer Numerical Control Machines (CNC)

CNCs are present in each workstation to process different parts. CNC machine are capable of being set up for different types of operations and on different part types which makes them functional in flexible manufacturing cell.

3.1.2 Industrial Robots

Industrial robot’s duty is to move products between conveyor and stations. It also moves products into and out of the manufacturing cell.

3.1.3 Conveyor

Conveyor functionality is to move parts between the stations and to bring finished products to exit point of the cell

3.1.4 Station PC

This part of system hardware’s task is to control the stations machine state of operation (whether machine is busy or in the idle state). It is also connected to station’s robot and conveyor and controls them. Station PCs are also connected to the PC which is connected to the reader, over the network to receive RFID data and to send stations’ state.

3.1.5 PLC

PLC is controlling the conveyor movement through orders it receives from stations.

3.1.6 RFID Reader

Reader collects data from tags through using antennas and provides this data to the PC it is connected to.

3.1.7 Reader Antennas

This piece of hardware sends the data to tag in the form of radio waves and returns data which come back from tags to the reader.

3.1.8 Tag

Tags are attached to parts and provide the identification code, and read and write ability of the data, so the system can detect parts.

3.1.9 PC Connected To Reader

This PC receives tag data from the RFID reader and by means of software system analyzes and separates useful data, then sends this data to the related station. It also receives a station state and show the real-time state of the system.

3.2 Software Design

Software suits are essential within the control system for it to be operational and are comprised of different parts.

First is the application software which is the software installed in each station, to control it by making proper decisions, based on the software internal logic and based on number and type of operations the station's machine can perform, and the data received about the part which is present at the station's entrance. Hence this software decides if the part can be entered to the station or not and is responsible for writing the data of the operations performed on part to it. This software must be able to interact and send control orders for industrial robots, CNC machine and conveyor. It also must be able to communicate to other software suits on the network to exchange data about the station and data related to tags.

The other application software which is used in the system is the monitoring software, which collects parts and stations states and provides real-time monitoring of the system to the user. This software shows which stations are busy or idle and which part type is under operation in the station and how much time is required for the process to finish and how long does it take for the station to be free for the next part.

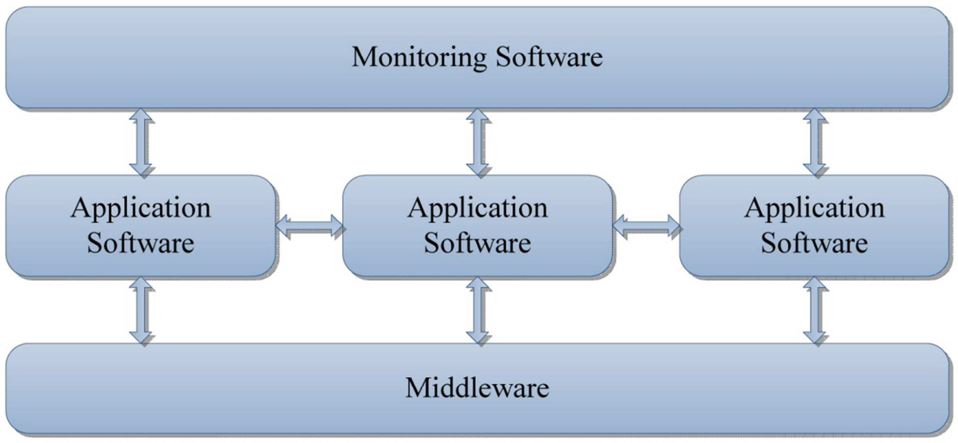


Figure 3.2**:** Software Suits within the Control System

The next important software in the control system is the software responsible for controlling RFID reader information, named middleware. Middleware in this system must be able to filter multi-tag readings. As each tag when is in the entrance of station will be read many times in every second, it is vital for the system to have a middleware to separate useful readings. Moreover, middleware must filter tags based on the antenna they are read from because each antenna is installed in one station and represents the entry point of that station. Software suits and their relations are shown in Figure 3.2.

3.3 System Behavior Description

When a part arrives into the flexible manufacturing cell entry point, reader detects it through antenna and will send its information to the middleware, middleware selects the first reading and ignores next readings of the tag and sends its data to the PC connected to the reader, at this point if part type is proper for the production plan of the cell, it evaluates part information and based on the state of stations(free or busy) and based on the type of parts and operations each station can process, puts the part on the conveyor to be sent to the desired station. If part type does not match cell production plant it will be put out of the entrance. In this way one of the requirements which were introduced during the system analysis phase which is preventing wrong production and incidents in production will be fulfilled and no wrong parts will enter the cell. The UML activity diagram for this process is shown in Figure 3.3.

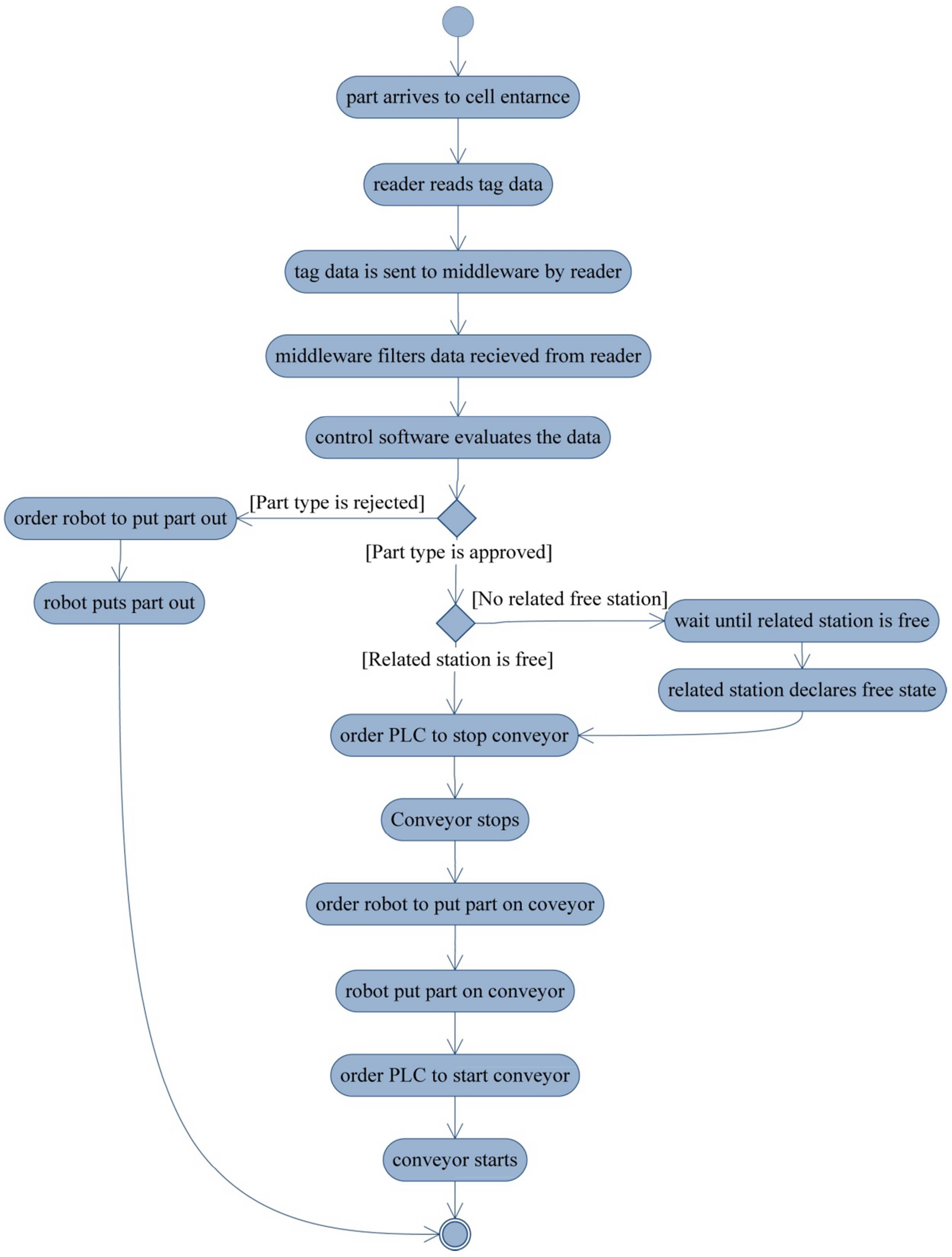

part arrives to cell entarnce
reader reads tag data
tag data is sent to middleware by reader
middleware filters data recieved from reader
control software evaluates the data
[Part type is rejected]
order robot to put part out
robot puts part out
[Part type is approved]
[No related free station]
wait until related station is free
related station declares free state
[Related station is free]
order PLC to stop conveyor
Conveyor stops
order robot to put part on coveyor
robot put part on conveyor
order PLC to start conveyor
conveyor starts

Figure 3.3**:** Process of Entering Part to Cell

When the part is on the conveyor, as it passes from stations entrance, reader sends part's data to the middleware and then middleware sends it to the same station, so if part type is not proper for the station, nothing will happen. But when it reaches to the desired station which can process the part, station will stop the conveyor and take the part. This last process would be done by control system of the station ordering conveyor to stop then order the industrial robot to pick up the part and put it in the CNC and after that ordering the conveyor to start again. The UML activity diagram for this process is shown in Figure 3.4 and Figure 3.5.

When the operation of part in each station finishes, the data on tag will be changed and the new state of part and information about operations done on it will be added to its tag memory, then station controller will stop conveyor put the part on it and start conveyor again then sends the state of the station (station is free and ready for new part) to the monitoring software and other stations over the network.

If this part needs other operations in the other workstations of the cell, the new information on its tag will make it possible for that station to detect part when it passes from the stations entrance and by the same process which is described, part will go to the station and required operations will be performed.

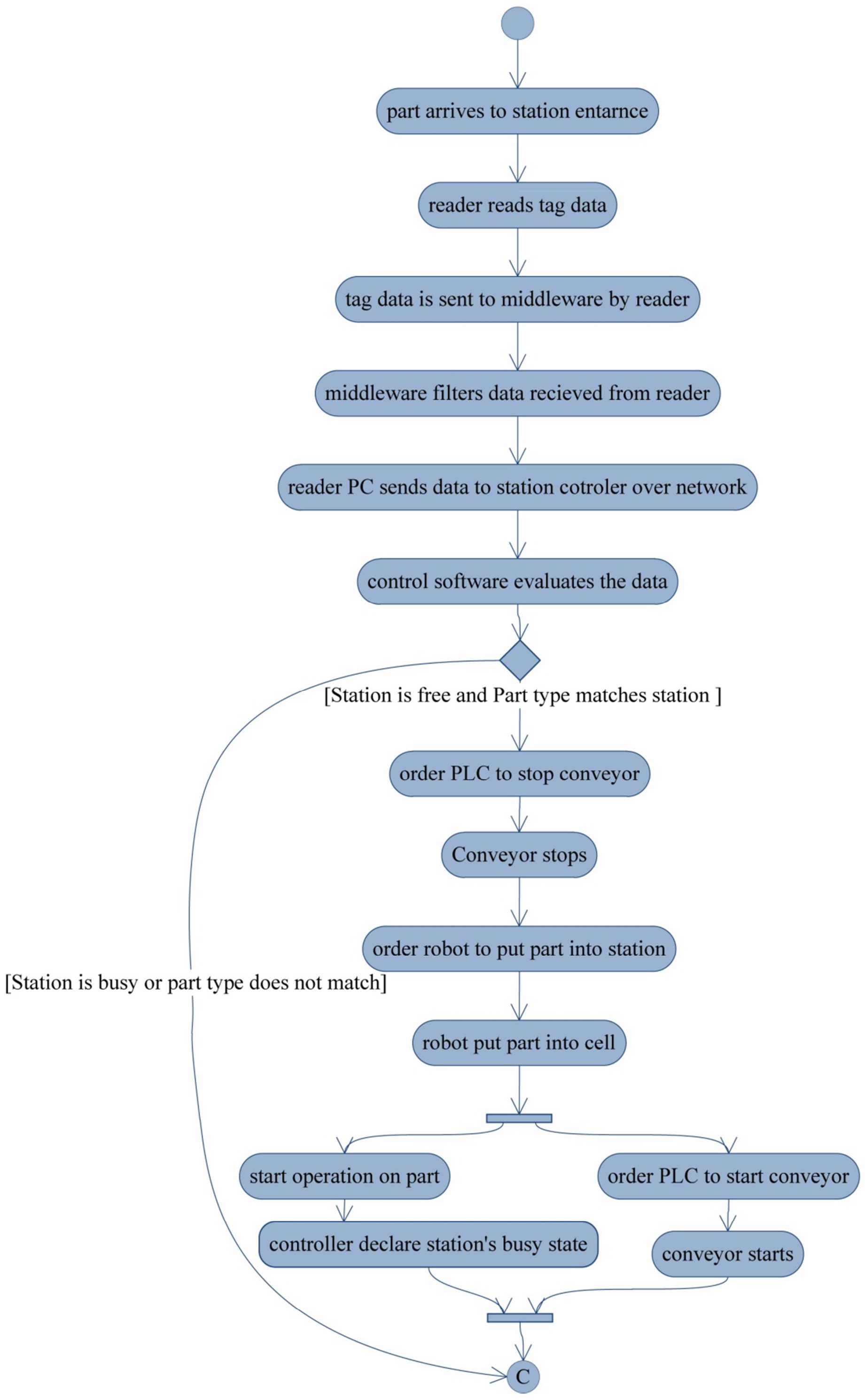

part arrives to station entarnce
reader reads tag data
tag data is sent to middleware by reader
middleware filters data recieved from reader
reader PC sends data to station cotroler over network
control software evaluates the data
[Station is free and Part type matches station]
order PLC to stop conveyor
Conveyor stops
order robot to put part into station
[Station is busy or part type does not match]
robot put part into cell
start operation on part
order PLC to start conveyor
controller declare station's busy state
conveyor starts
C

Figure 3.4: Process of Entering Part to Station

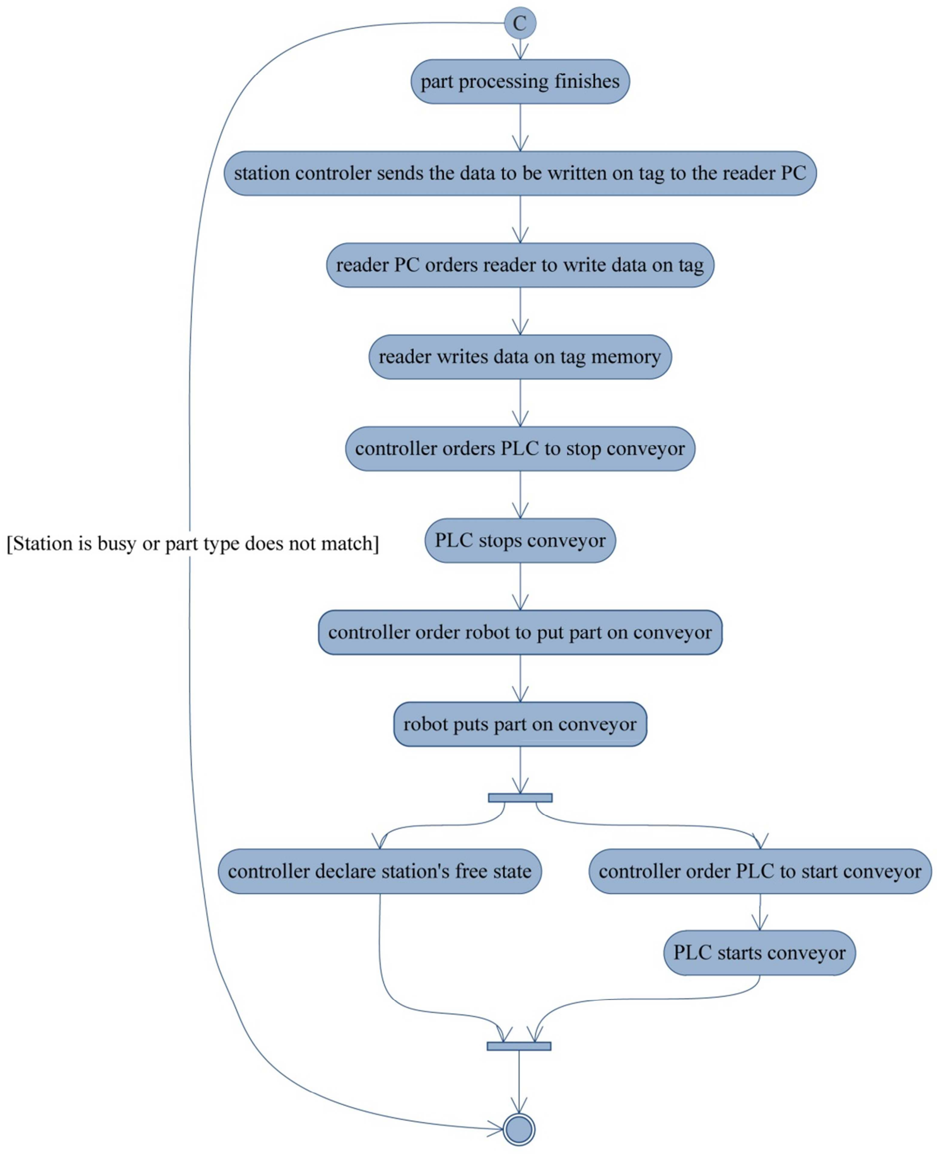

C
part processing finishes
station controler sends the data to be written on tag to the reader PC
reader PC orders reader to write data on tag
reader writes data on tag memory
controller orders PLC to stop conveyor
[Station is busy or part type does not match]
PLC stops conveyor
controller order robot to put part on conveyor
robot puts part on conveyor
controller declare station's free state
controller order PLC to start conveyor
PLC starts conveyor

Figure 3.5: Process of Entering Part to Station

If a finished product or part reaches to the exit of the cell it would be put out of the manufacturing cell by the same manner mentioned earlier. The UML activity diagram for this process is shown in Figure 3.6.

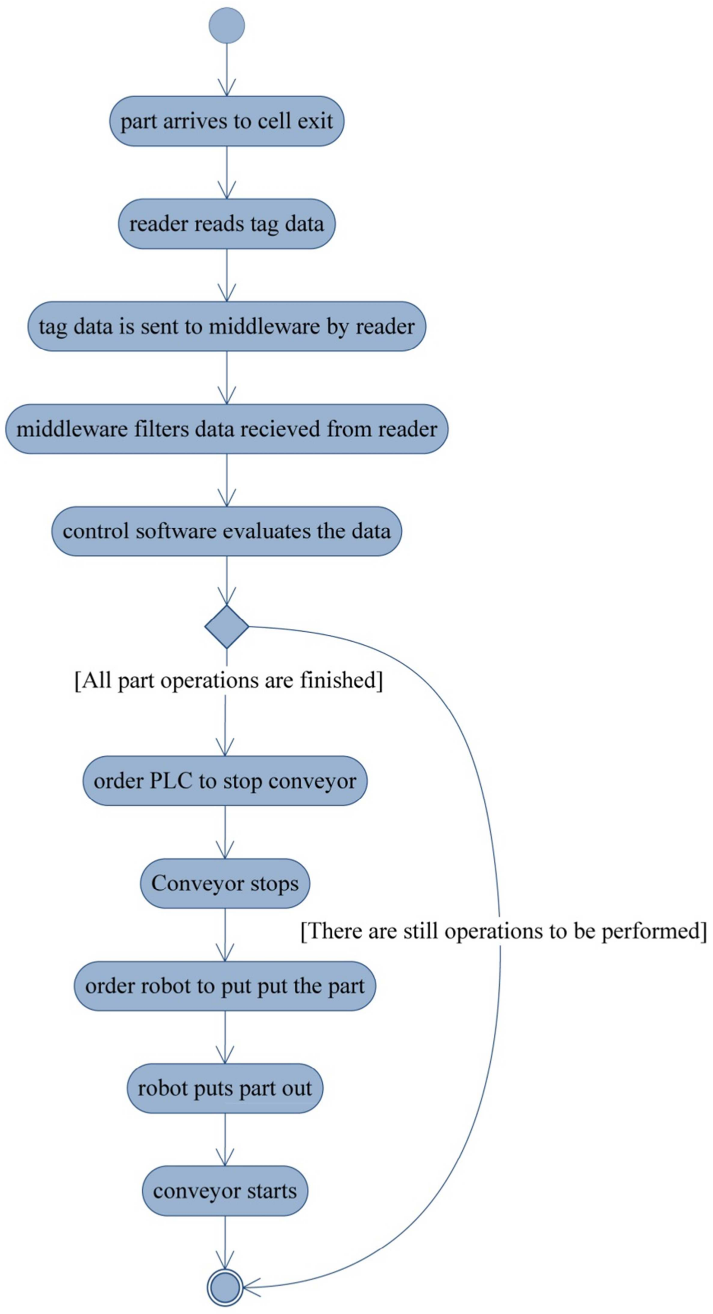

part arrives to cell exit
reader reads tag data
tag data is sent to middleware by reader
middleware filters data recieved from reader
control software evaluates the data
[All part operations are finished]
order PLC to stop conveyor
Conveyor stops
[There are still operations to be performed]
order robot to put put the part
robot puts part out
conveyor starts

Figure 3.6: Process of Exiting Part from Cell

For the reason that in time of all of the operations the state of part and the station is sent to the monitoring software, real-time monitoring is enabled through the system design.

Furthermore, as a result of writing data on part, that each operation possibility of tracking errors and faults in the production system is provided and quality control is made simpler and faster.

Since part types can be detected in the system and each part will go to related station, there is no fixed input order of parts to the cell. Parts from different types can enter cell and will go through their planned operation sequence and proper stations automatically.

Another requirement to be satisfied by this system was adding customization ability to the system. By defining and writing the proper code in the part’s tag memory, including information about the specific operations to be done, part goes through those specific operations and any customization on part will be obtainable.

Even real-time customization is possible by writing the new information about customized operations on tag during the production and between different operations. For instance, if some parts are decided to have other features after they have gone under some operations, new information can be written on part’s tag to go under new operations.

As it can be seen all system requirements indicated in the system analysis phase can be reached. These results have been verified by implementing the system. Implementation is discussed in the next chapter.

Chapter 4

IMPLEMENTATION

In the implementation phase, the proposed control system is developed and tested to exhibit advantages and benefits of RFID enabled control for a manufacturing cell and display the behavior of control system in practice. Implementation consists of two phases, hardware design and execution and software development. These phases are explained in detail in this chapter.

Initially the layout of the implemented system, hardware and software systems are described and then the scenario which is implemented is presented. The flexible manufacturing cell layout is shown in Figure 4.1 including the hardware connections.

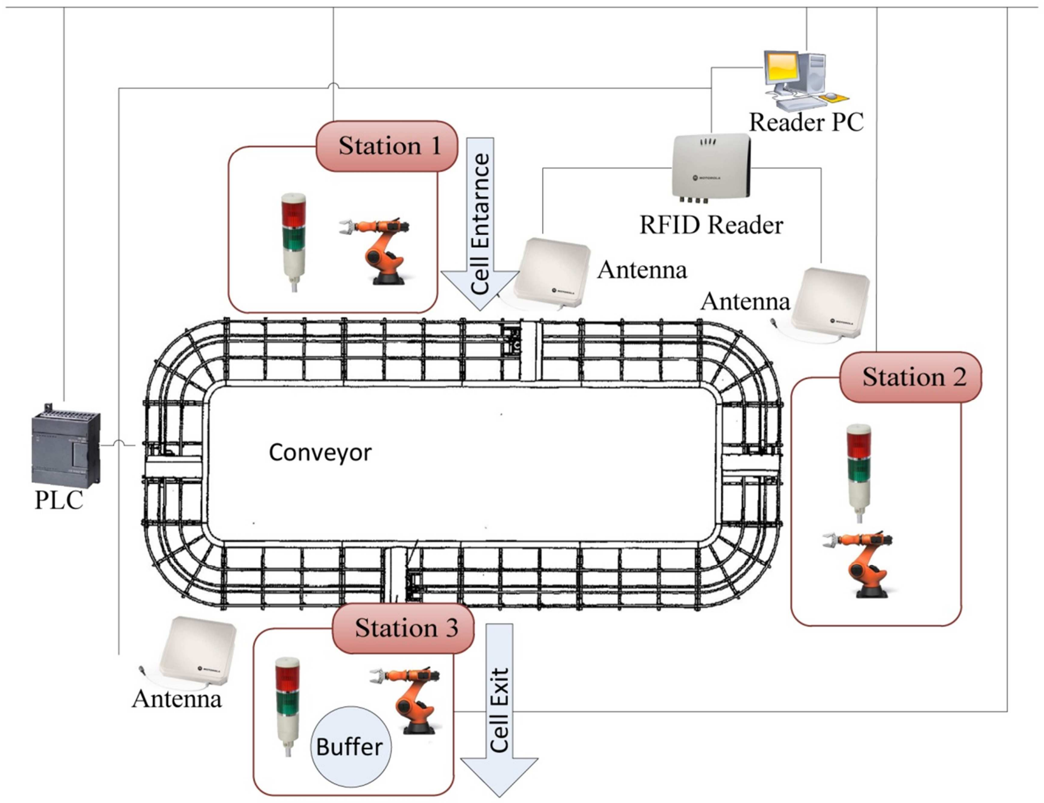


Figure 4.1: Implemented RFID Enabled Manufacturing Cell Layout

Based on the availability of resources like CNC machines in the computer integrated manufacturing laboratory and considering the fact that it is not required to have PC for each station when there is no CNC in each station to be controlled, system hardware architecture is altered to the one in the Figure 4.1. Instead of CNC machines and to show the state of each station machine, a LED is utilized, and the application software of all stations is installed in the same PC which is connected to RFID reader. The entrance of the cell is located beside station number one and robot in this station is responsible for inputting parts to the cell. The exit of the cell also is located beside station number three and robot of the station number three is responsible to put part out of the cell.

There is no change in holistic procedure of system actions and this architecture clearly presents the benefits and advantages of control system designed.

First phase of implementation is hardware phase. Hardware components, their functionality and specifications and their connections are described.

4.1 Hardware Development

Some basic hardware components like PC connected to the reader, LEDs and conveyor are conventional hardware which does not require detailed description.

The RFID reader installed is Motorola FX7400 fixed RFID reader which is shown in Figure4.2. Its communications capability is over 10/100 Baset Ethernet (rJ45) w/ PoE support and also USB Client (USB type B). It has 2 inputs and 2 outputs General Purpose port which are optically isolated (terminal Block). Its Power Supply is +24Vdc or PoE (IEEE 802.3af) and it has 4 mono-static antenna ports (reverse Polarity tNC) and the Frequency (UHF Band) of Reader is 902 MHz~928 MHz, 865 MHz~868 MHz with the Power Output of +15dBm to +30dBm. Its IP addressing can be Static and Dynamic. Application programming interface it supports are .NET and C.

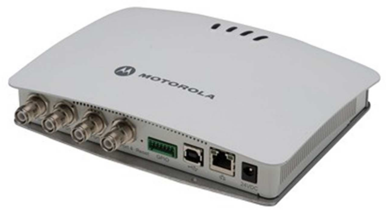


Figure 4.2: Motorola FX7400 fixed RFID Reader

RFID antennas are made by Alien Company (shown in Figure4.3) and their frequency range is 865 - 965 MHz with circular polarization and gain of 5.5dBiL max. Their cable is 6 Meters LMR-195, 50 ohm coaxial with reverse polarity connector. Their dimensions are 25cm x 25cm x 3.8cm.

Figure 4.3: Alien ALR-8610-AC

Tags used are Confidex Steelwave Micro passive UHF RFID transponder, working with 865-928 MHz frequency with 96 bit EPC memory. Their read range is up to 3 meters when they are installed on metal. This tag is shown in Figure 4.4.

Figure 4.4: Confidex Steelwave Micro Passive UHF RFID Transponder

PLC implemented is Siemens Simatic s7-200 with 8 analogue input and 6 output. It can be set-up by AC or DC power source and it is shown in Figure 4.5.

Figure 4.5: Siemens Simatic s7-200 PLC

There are 2 kind of robots used. First is EduBot 100 SC which has 5 axes, plus 1 gripper (suction type), each with 270 motion range with accuracy of 2 mm and total vertical reach of 475 mm and total horizontal reach of 345 mm. and is connected via RS232 serial port. The second type is EduBot 250 S which also has 5 axis, plus 1 gripper, but the gripper is also servo motor driven, each with 270 motion range with accuracy of 2 mm and total vertical reach of 475 mm and total horizontal reach of 345 mm. and is connected via RS232 serial port. These Robots are shown in Figure4.6.

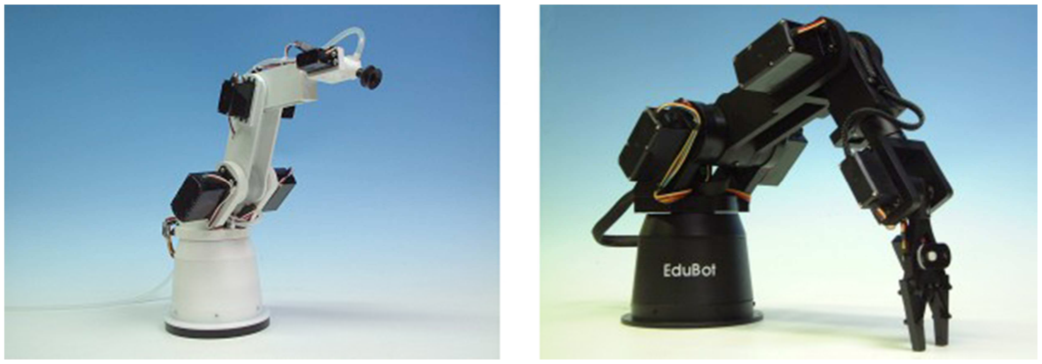


Figure 4.6: EduBot 100 SC and EduBot 250 S Robot Arms

To connect PC to PLC hardware interface is designed and built for this control system which translates the output of PC's parallel port to analog input of PLC. The electronic circuit design based on the output of LPT ports and required 24 voltage DC input of PLC is designed and developed to fit requirements of control system. The hardware interface is shown in Figure4.7.

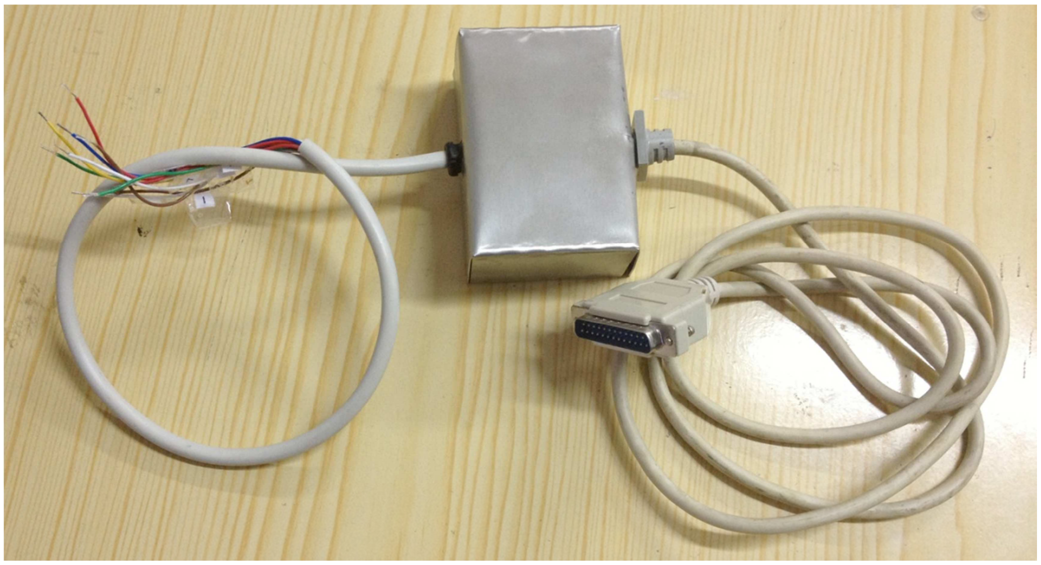


Figure 4.7: PC to PLC Hardware Interface

All hardware components have been installed in computer integrated manufacturing laboratory and connections are made as illustrated in Figure 4.1.

4.2 Software Development

Second phase of implementation is software development. Software development itself is divided to planning and implementing phases. In the planning phase there have to be a requirement analysis for the software development, or in other words, what are the functionalities which software must perform. The requirements of software developed for this control system is the ability to interact with external systems and controllers, likes robots, PLC and reader. Moreover it needs to possess a graphical user interface (GUI) to provide the real-time system monitoring and to let the user run the application and give required commands. Software also must contain internal logic of the control mechanism to conduct proper orders to external systems. The internal logic uses data coming from external systems and sends them resulted commands. Software architecture of control system is shown in Figure 4.8.

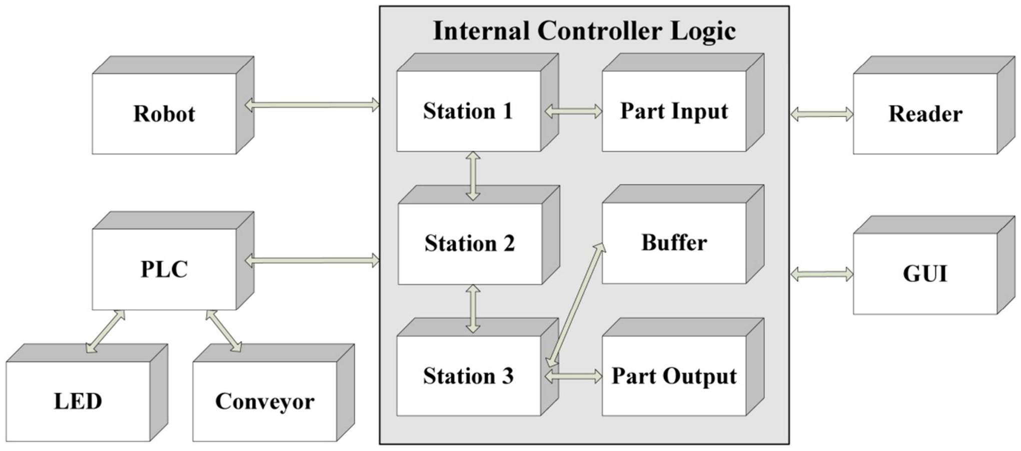


Figure 4.8: Software Architecture of Control System Implemented

As it is shown in software architecture, programming language must have the ability of connecting to reader, robots and PLC. It is mentioned in hardware specifications that the reader used in this control system can be controlled by .NET software collection and C#. C# is one of the powerful modern object-oriented programming languages which is the best fit for this work. The internal controller part can communicate to robots ordering them to move every axis to the desired direction and angle. It also can send control commands to PLC for the movement of conveyor and turning on and off LEDs. PLC is programmed beforehand using STEP7 Micro/Win software to control conveyor and LEDs based on the receiving order from C#.

Internal controller is able to connect to the reader and send commands to start reading, configuring reader antennas, and writing proper data on tags.

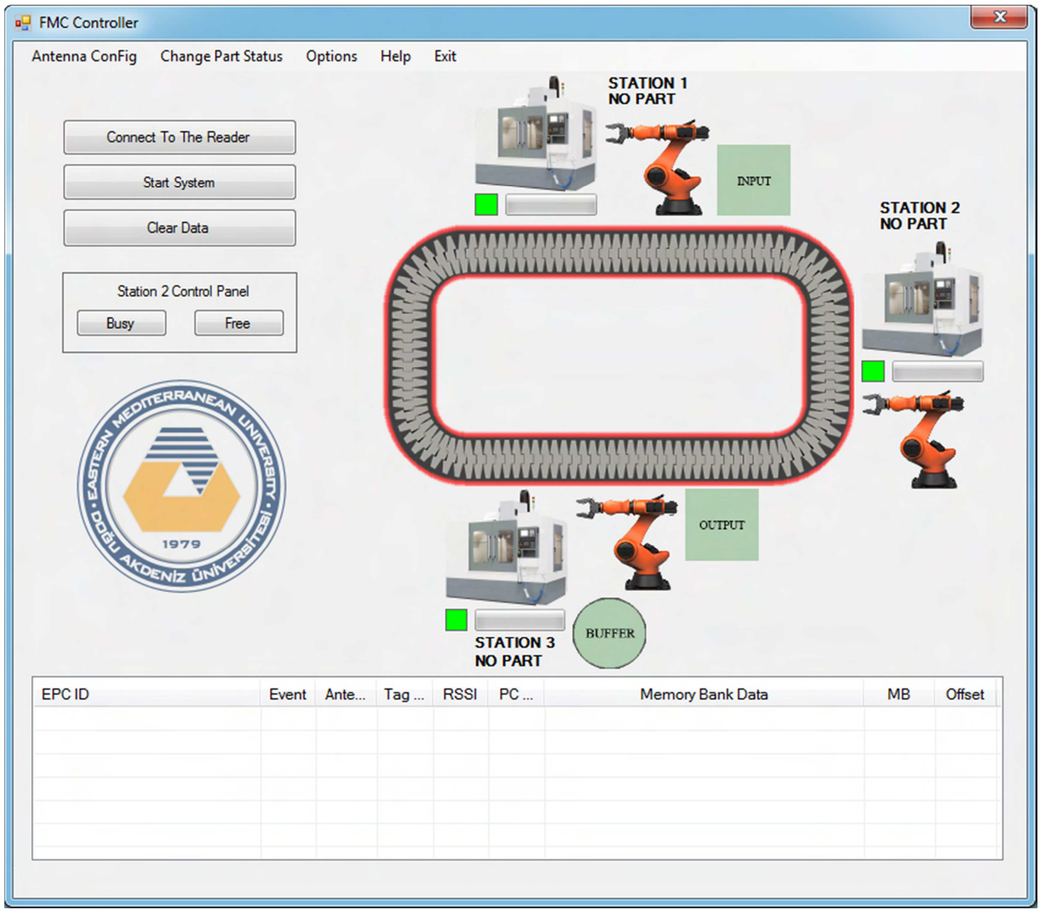
Software of each station is capable of making decisions based on the type of tag, to perform proper operation through communicating with reader, PLC and robots. It also sends data to be written on tag. It sends the station’s state to the graphical user interface (GUI).

Part input software piece is able to communicate to reader to get tag data, communicate to the stations’ software to be informed of their states, then makes best decision to put part into the cell for the system to work optimally.

Part output software also is capable of communicating with reader, PLC and robots to put finished parts out of the system.

Buffer software is capable of communicating with station number three software , reader, PLC and robots to take parts from station number three when the station that part should go to are busy.

Graphical user interface (GUI) of the software is divided into four areas and is shown in Figure 4.9. Menu bar which contains: antenna configuration, change part status, options, help and exit.



EPC ID	Event	Ante...	Tag ...	RSSI	PC ...	Memory Bank Data	MB	Offset

Figure 4.9: FMC Controller GUI

At antenna configuration the transmission power of each antenna can be adjusted to adjust the reading range of each antenna to fit system. In change part status, data can be written on tags to change their part type. In the option menu there are three choices to show or hide, tag readings (which is the table at the lower part of GUI), status message (which shows messages software sends to some of the order sends by the user), and station control which gives the choice to control the station number two manually. Finally, help provides some information about the software.

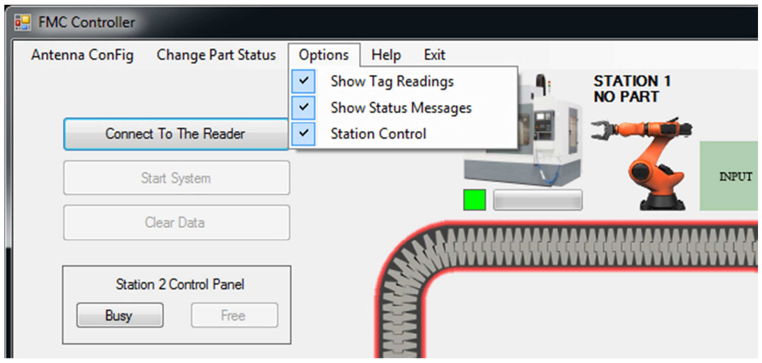


Figure 4.10: GUI Option Menu

Control buttons, which perform basic software operations, are the next part in GUI. First button connects the system to RFID reader. If reader is successfully connected, the text of bottom will be changed to "disconnect from reader". The start system button starts the flexible manufacturing cell operations and by pressing this button system waits for parts to arrive at input of the system to perform other actions. Clear data button deletes data in the software about the parts produced in previous system run.

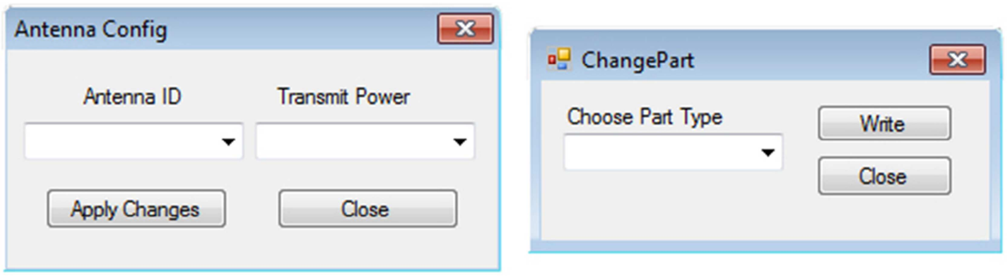


Figure 4.11: GUI Antenna Config and Changepart Windows

Station number two control panel lets user make the station number two busy or free manually to see the reaction of the system to disturbances. The monitoring part of GUI shows the system hardware layout and their states. Every station is shown and its state is shown by a green or red box next to it. There is a progress bar in each station which shows how much work is done in each operation and when the work on one part would be finished. The part type which is inside each station is also written in GUI. The conveyor state (moving or stopped) is also shown by the green or red color around it. Input, output and buffer zone in station number three, are also shown in monitoring part of GUI. The tag reading part of GUI shows the detailed information of tags read during the manufacturing cell operation. This part also provides administrative actions performed on tags in necessary occasions, like if user wants to define new tags for the system.

4.3 Scenario

To observe system functionality a scenario is devised for the flexible manufacturing cell which describes operations in the stations, part types, number of the stations and other manufacturing related specification.

Three part types are defined to be produced in the cell, namely part A, part B and part C. Each of these three parts has different production plans and must go through different processes with different operation times to reach to the desired finished product. In the same manner, stations have different capabilities in performing operations needed for parts. Part A is considered to be processed during one sequence of operations and all the stations are considered to be able to perform those operations, thus if a part of type A enters any of the stations it will be processed thoroughly and would get to the finished state afterwards. Part C is also considered to be processed during one sequence of operations but only station number two and station number three are capable of performing those sequence of operations. So if a part of type C enters the station number two or station number three it would be processed thoroughly and get to the finished states afterwards. Part B is considered to be processed during two sequences of operations. The first sequence is considered to be performed in either station number two or station number three and the second sequence is considered to be performed in either station number one or station number two. If part B enters station number three for the first sequence of operations it has to enter one of the other stations for the second sequence of operations and then parts B is at finished state. But if a part B enters station number two which is capable of performing both sequences of operations on part B, then parts B will be thoroughly processed in station number two and will be at finished states afterwards. Figure 4.4 shows part types and stations ability to perform operations on them.

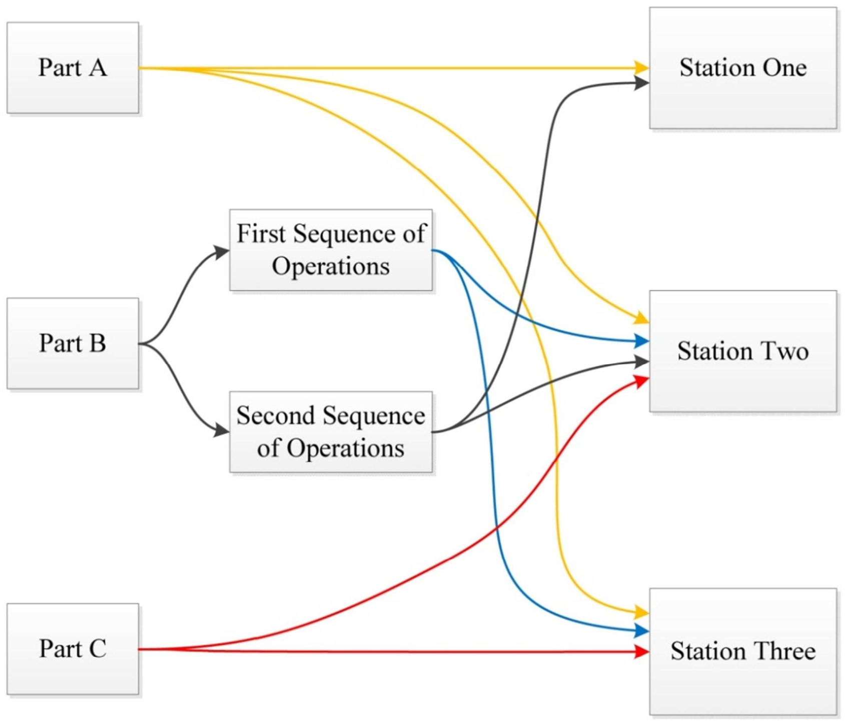


Figure 4.12: Part Types and Stations Capabilities in Performing Operation

When software is run and user pushes the “Connect To The Reader” button and reader gets connected, then “Start System” button can be pushed and system will wait for a part to enter to the entrance zone. After a part of defined type (A, B or C) arrives to the entrance (if other parts arrive they will be put out), controller decides which station is best choice to produce this part based on the least time spent on the conveyor and type of operations stations can perform, and then sends the part to that station. For the next parts coming to the entrance, system also checks the availability of stations, not to send a part to a busy station. In each station if the part production is finished and part is at finished state, stations put them on conveyor to go to exit and be put out. If part is of type B and the second sequence of its operations is left, then station’s controller checks if it can be processed at the moment based on the availability of other stations or not. If other stations capable of performing second sequence are busy, part will be put to buffer and as soon as one of them is free, part will be sent to that station.

In the same way described above, if parts enter the cell with any order, system automatically make decisions and perform the operations. To show the way system works one possible case of cell operation is studied.

The first part which enters the cell is of type A and since the shortest route to a station which is capable of performing this part’s operations is to station number one, system sends part to station number one. Next part is of type B that comes to the system entrance and since station number two is capable of performing both sequences of this part’s operations, part is sent to station number two. Next part is of type C and since only station number three is free and it is able to work on part, then part is sent to station number three. After that a part of type B is received to the cell entrance and since all stations are busy it will wait until station number three finishes its operation and then part is sent to that station. Station number three can only perform first sequence of operations of part type B, hence when it finishes this sequence, controller check for the availability of other stations, meanwhile station one has finished the operation on part type A and is free, so station number three sends part type B to station number one for the second sequence of operations. When each station finishes its operation it sends the part to the exit. Figure 4.13 shows these steps.

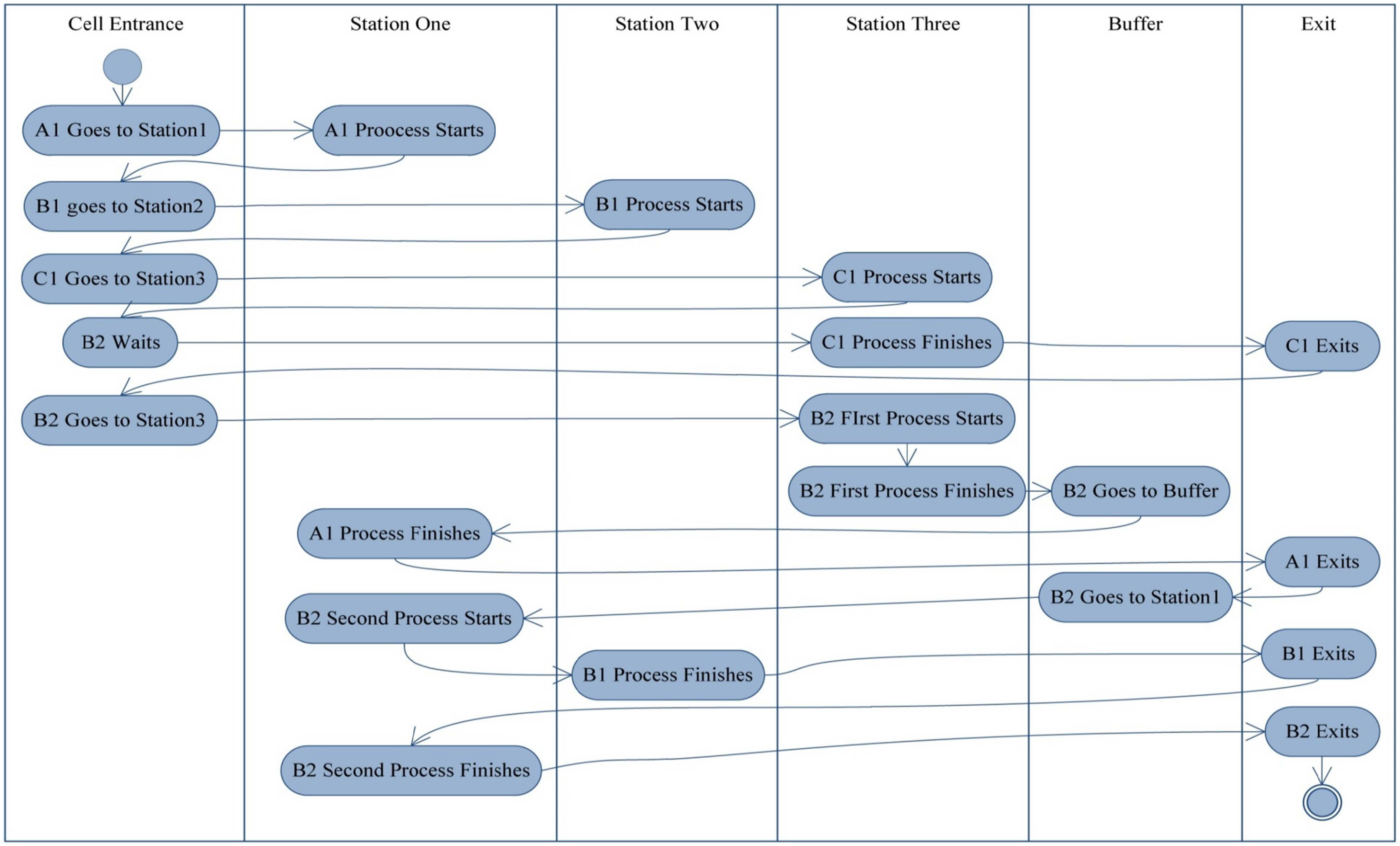


Figure 4.13: Activity Diagram of Studied Case

In the proposed and implemented system, if one of the stations stops working for any reason, for instance incidents during the production or even machine maintenance, other stations are capable of continuing the production of all part types and system automatically distribute parts between two other stations. This feature can be observed through making station number two busy manually from the software GUI. The changes in routings of parts in the previous studied case based on making station number two busy are shown at Figure 4.14.

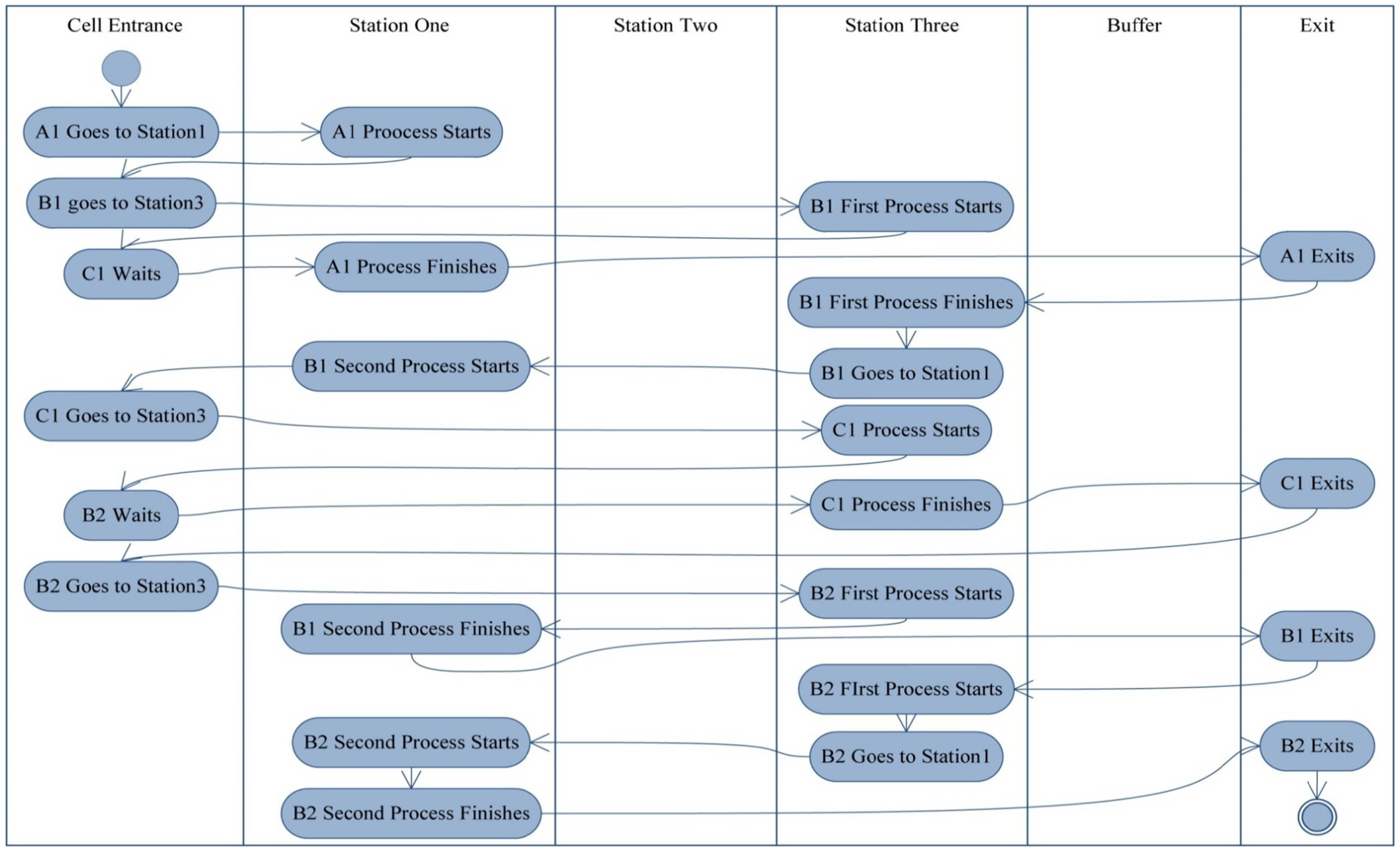


Figure 4.14: Activity Diagram of Changed Studied Case

Chapter 5

CONCLUSION

For organizations to remain in the market competition, they have to possess flexibility in manufacturing. Current control systems lack enough flexibility to adapt to present market conditions. This dissertation has investigated application of RFID into manufacturing control system to increase the flexibility while maintaining productivity of the manufacturing system and the quality of the production.

This project was undertaken to design and develop a RFID enabled flexible manufacturing cell control system to overcome deficiencies existing in current FMC control mechanisms. The results of this implementation have shown the control system proposed provides various benefits including bringing the real-time monitoring ability to the system, facilitating quality control and fault finding based on the unique identification code of each part, reducing errors in management of raw material handling, production operations and finished products, enabling product customization, providing capability of reacting to incidents at production time, creating the chance of real-time customization of products and hence practicality and improvement of just in time manufacturing paradigm and at last providing capability of simultaneous multiple type production with no predefined order.

REFERENCES

Beckner, M., Simms, M., & Venkatesh, R. (2009). Pro RFID in BizTalk Server 2009: Apress.

Bolic, M., Simplot-Ryl, D., & Stojmenovic, I. (2010). RFID systems: research trends and challenges: Wiley. com.

Bongaerts, L., Monostori, L., McFarlane, D., & Kádár, B. (2000). Hierarchy in distributed shop floor control. Computers in Industry, 43(2), 123-137.

Browne, J., Dubois, D., Rathmill, K., Sethi, S. P., & Stecke, K. E. (1984). Types of flexibilities and classification of flexible manufacturing systems. The university of Michigan.

Chan, D.-Y., & Bedworth, D. D. (1990). Design of a scheduling system for flexible manufacturing cells†. The international journal of production research, 28(11), 2037-2049.

Chen, F. F., & Adam Jr, E. E. (1991). The impact of flexible manufacturing systems on productivity and quality. Engineering Management, IEEE Transactions on, 38(1), 33-45.

Cheong, T.-S., & Lee, Y.-J. RFID Information Value Chain and ETRI RFID Ecosystem: Value-added Environment Linking Physical and Virtual Worlds. Development and Implementation of RFID Technology, ISBN, 978-973.

Dilts, D., Boyd, N., & Whorms, H. (1991). The evolution of control architectures for automated manufacturing systems. *Journal of manufacturing systems, 10*(1), 79-93.

Hagl, A., & Aslanidis, K. (2009). RFID: Fundamentals and applications *RFID Security* (pp. 3-26): Springer.

Hua, J. W., Sun, H. X., Liang, T., & Lei, Z. M. (2008). The Design of Manufacturing Execution System Based On RFID. *2008 Workshop on Power Electronics and Intelligent Transportation System, Proceedings*, 8-11. doi: 10.1109/peits.2008.72

Huang, G. Q., Zhang, Y. F., & Jiang, P. Y. (2007). RFID-based wireless manufacturing for walking-worker assembly islands with fixed-position layouts. *Robotics and Computer-Integrated Manufacturing, 23*(4), 469-477. doi: http://dx.doi.org/10.1016/j.rcim.2006.05.006

Johnson, D. (2002). RFID tags improve tracking, quality on Ford line in Mexico. *Control Engineering, 49*(11), 16-16.

Kohn, W., Brayman, V., & Littleton, J. (2005). Repair-control of enterprise systems using RFID sensory data. *Iie Transactions, 37*(4), 281-290. doi: 10.1080/07408170590516953

Liu, M. R., Zhang, Q. L., Ni, L. M., & Tseng, M. M. (2004). An RFID-Based distributed control system for mass customization manufacturing. In J. Cao, L. T.

Yang, M. Guo & F. Lau (Eds.), *Parallel and Distributed Processing and Applications, Proceedings* (Vol. 3358, pp. 1039-1049). Berlin: Springer-Verlag Berlin.

McFarlane, D., Sarma, S., Chirn, J. L., Wong, C. Y., & Ashton, K. (2003). Auto ID systems and intelligent manufacturing control. *Engineering Applications of Artificial Intelligence, 16*(4), 365-376. doi: http://dx.doi.org/10.1016/S0952-1976(03)00077-0

Miles, S. B., Sarma, S. E., & Williams, J. R. (2008). *RFID technology and applications* (Vol. 1): Cambridge University Press Cambridge.

Sethi, A. K., & Sethi, S. P. (1990). Flexibility in manufacturing: a survey. *International journal of flexible manufacturing systems, 2*(4), 289-328.

Shivanand, H. (2006). *Flexible manufacturing system*: New Age International.

Sun, H. (2011). Applications of RFID Technology in the Complex Product Assembly Executive Process *Designing and Deploying RFID Applications*.

Upton, D. M. (1995). Flexibility as process mobility: the management of plant capabilities for quick response manufacturing. *Journal of Operations Management, 12*(3), 205-224.

Wong, T., & Leung, C. (2010). *Application of Auto-ID in agent-based manufacturing control*. Paper presented at the Computers and Industrial Engineering (CIE), 2010 40th International Conference on.

ZELENOVIĆ, D. M. (1982). Flexibility—a condition for effective production systems. *The international journal of production research, 20*(3), 319-337.

APPENDICES

Appendix A: Write Data on Tag Code

```
namespace Majid_RFID_Project
{
    public partial class ChangePart : Form
    {
        private AppForm m_AppForm;
        internal TagAccess.WriteAccessParams m_WriteParams;
        string thistagidforhere;
        internal WriteProblemForm m_WriteProblemForm;
        internal WriteSuccessfulForm m_WriteSuccessfulForm;

        internal RFIDReader m_ReaderAPI;
        internal AntennaInfoForm m_AntennaInfoForm;


        public ChangePart(AppForm appForm, AntennaInfoForm
m_AntennaInfoForm, RFIDReader m_ReaderAPI)
        {
            InitializeComponent();

            m_AppForm = appForm;
            this.m_ReaderAPI = m_ReaderAPI;
            this.m_AntennaInfoForm = m_AntennaInfoForm;

            m_WriteParams = new TagAccess.WriteAccessParams();
            m_WriteParams.MemoryBank = MEMORY_BANK.MEMORY_BANK_EPC;
            m_WriteParams.AccessPassword = 0;
            m_WriteParams.ByteOffset = 14;
            m_WriteParams.WriteDataLength = 2;
            comboBox1.SelectedIndex = 0;
            thistagidforhere = "";
            m_WriteProblemForm = new WriteProblemForm();
            m_WriteSuccessfulForm = new WriteSuccessfulForm();
        }
        private void WriteButton_Click_1(object sender, EventArgs e)
        {
            try
            {
                m_WriteParams = new TagAccess.WriteAccessParams();
                m_WriteParams.MemoryBank =
MEMORY_BANK.MEMORY_BANK_EPC;
                m_WriteParams.AccessPassword = 0;
                m_WriteParams.ByteOffset = 14;
                m_WriteParams.WriteDataLength = 2;

                byte[] writeData = new
byte[m_WriteParams.WriteDataLength];
                if (comboBox1.SelectedIndex == 0)
                {
                    thistagidforhere = "1A00";
                }
                else if (comboBox1.SelectedIndex == 1)
                {
                    thistagidforhere = "2A00";
                }
                else if (comboBox1.SelectedIndex == 2)
                {
                    thistagidforhere = "3A00";
                }
                else if (comboBox1.SelectedIndex == 3)
                {
                    thistagidforhere = "4A00";
                }
                else if (comboBox1.SelectedIndex == 4)
                {
                    thistagidforhere = "5A00";
                }
                else if (comboBox1.SelectedIndex == 5)
                {
                    thistagidforhere = "1B00";
                }
                else if (comboBox1.SelectedIndex == 6)
                {
                    thistagidforhere = "2B00";
                }
                else if (comboBox1.SelectedIndex == 7)
                {
                    thistagidforhere = "3B00";
                }
                else if (comboBox1.SelectedIndex == 8)
                {
                    thistagidforhere = "4B00";
                }
                else if (comboBox1.SelectedIndex == 9)
                {
                    thistagidforhere = "5B00";
                }
                else if (comboBox1.SelectedIndex == 10)
                {
                    thistagidforhere = "1C00";
                }
                else if (comboBox1.SelectedIndex == 11)
                {
                    thistagidforhere = "2C00";
                }
                else if (comboBox1.SelectedIndex == 12)
                {
                    thistagidforhere = "3C00";
                }
                else if (comboBox1.SelectedIndex == 13)
                {
                    thistagidforhere = "4C00";
                }
                else if (comboBox1.SelectedIndex == 14)
                {
                    thistagidforhere = "5C00";
                }
                for (int index = 0; index < 
m_WriteParams.WriteDataLength; index += 2)
                {
                    writeData[index] = 
byte.Parse(thistagidforhere.Substring(index * 2, 2), 

System.Globalization.NumberStyles.HexNumber);
                    writeData[index + 1] = 
byte.Parse(thistagidforhere.Substring((index + 1) * 2, 2),

System.Globalization.NumberStyles.HexNumber);
                }

                m_WriteParams.WriteData = writeData;
                if (comboBox1.SelectedIndex == 0)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200498000000000000001A01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200498000000000000001A02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 1)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000002A01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000002A02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 2)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000003A01",
m_WriteParams, m_AntennaInfoForm.getInfo());
```

```
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000003A02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 3)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000004A01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000004A02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 4)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000005A01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000005A02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
```

```
                        m_WriteProblemForm.ShowDialog();
                    }
                }
            }
            else if (comboBox1.SelectedIndex == 5)
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                        "2005630000000000000001B02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                    m_WriteSuccessfulForm.ShowDialog();
                }
                catch
                {
                    m_WriteProblemForm.ShowDialog();
                }
            }
            else if (comboBox1.SelectedIndex == 6)
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                        "2005630000000000000002B02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                    m_WriteSuccessfulForm.ShowDialog();
                }
                catch
                {
                    m_WriteProblemForm.ShowDialog();
                }
            }
            else if (comboBox1.SelectedIndex == 7)
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                        "2005630000000000000003B02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                    m_WriteSuccessfulForm.ShowDialog();
                }
                catch
                {
                    m_WriteProblemForm.ShowDialog();
                }
            }
            else if (comboBox1.SelectedIndex == 8)
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                        "2005630000000000000004B02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                    m_WriteSuccessfulForm.ShowDialog();
                }
                catch
                {
                    m_WriteProblemForm.ShowDialog();
                }
            }
            else if (comboBox1.SelectedIndex == 9)
```

```
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000005B02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        m_WriteProblemForm.ShowDialog();
                    }
                }
                else if (comboBox1.SelectedIndex == 10)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000001C01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000001C02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 11)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000002C01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000002C02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 12)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "20056300000000000000003C01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "20056300000000000000003C02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 13)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "20056300000000000000004C01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "20056300000000000000004C02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
                else if (comboBox1.SelectedIndex == 14)
                {
                    try
                    {
                        m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "20056300000000000000005C01",
m_WriteParams, m_AntennaInfoForm.getInfo());
                        m_WriteSuccessfulForm.ShowDialog();
                    }
                    catch
                    {
                        try
                        {
                            m_ReaderAPI.Actions.TagAccess.WriteWait(
                            "200563000000000000005C02",
m_WriteParams, m_AntennaInfoForm.getInfo());
                            m_WriteSuccessfulForm.ShowDialog();
                        }
                        catch
                        {
                            m_WriteProblemForm.ShowDialog();
                        }
                    }
                }
            }
            catch
            {

            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}
```

Appendix B: Station Three Code for Part Type B

```
private void S3B1BackgroundWorker_DoWork(object sender, DoWorkEventArgs e)
        {
            if (Conveyor_Status == "START")
            {
                Conveyor_Status = "STOP";
                LPTPort = Conveyor1.Stop(LPTPort);
                Robot3.PutInS3();
                Thread.Sleep(3000);
            }
            else
            {
                while (true)
                {
                    if (Conveyor_Status == "START")
                    {
                        Conveyor_Status = "STOP";
                        LPTPort = Conveyor1.Stop(LPTPort);
                        Robot3.PutInS3();
                        Thread.Sleep(3000);
                        break;
                    }
                }
            }
            LPTPort = Conveyor1.Start(LPTPort);
            Conveyor_Status = "START";

            string thistagid = "";

            m_WriteParams = new TagAccess.WriteAccessParams();
            m_WriteParams.MemoryBank = MEMORY_BANK.MEMORY_BANK_EPC;
            m_WriteParams.AccessPassword = 0;
            m_WriteParams.ByteOffset = 14;
            m_WriteParams.WriteDataLength = 2;
            byte[] writeData = new byte[m_WriteParams.WriteDataLength];
            if (tagIDforwritingS3B0 == "200563000000000000001B00")
            {
                thistagid = "1B01";
            }
            else if (tagIDforwritingS3B0 == "200563000000000000002B00")
            {
                thistagid = "2B01";
            }
            else if (tagIDforwritingS3B0 == "200563000000000000003B00")
            {
                thistagid = "3B01";
            }
            else if (tagIDforwritingS3B0 == "200563000000000000004A00")
            {
                thistagid = "4B01";
            }
            else if (tagIDforwritingS3B0 == "200563000000000000005B00")
            {
                thistagid = "5B01";
            }

            for (int index = 0; index < m_WriteParams.WriteDataLength; index
+= 2)
```

```
            {
                writeData[index] = byte.Parse(thistagid.Substring(index * 2,
2),
                    System.Globalization.NumberStyles.HexNumber);
                writeData[index + 1] = byte.Parse(thistagid.Substring((index +
1) * 2, 2),
                    System.Globalization.NumberStyles.HexNumber);
            }

            m_WriteParams.WriteData = writeData;
            if (m_ReaderAPI.Actions.TagAccess.OperationSequence.Length > 0)
            {

m_ReaderAPI.Actions.TagAccess.OperationSequence.StopSequence();
            }
            else
            {
                m_ReaderAPI.Actions.Inventory.Stop();
            }
            if (tagIDforwritingS3B0 == "2005630000000000000001B00")
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                        "2005630000000000000001B00", m_WriteParams,
m_AntennaInfoForm.getInfo());
                    tagIDforwaitongnextstep = "2005630000000000000001B01";
                }
                catch (Exception)
                {
                    MessageBox.Show("Could Not Write On Part");
                }
            }
            else if (tagIDforwritingS3B0 == "2005630000000000000002B00")
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                         "2005630000000000000002B00", m_WriteParams,
m_AntennaInfoForm.getInfo());
                    tagIDforwaitongnextstep = "2005630000000000000002B01";
                }
                catch (Exception)
                {
                    MessageBox.Show("Could Not Write On Part");
                }
            }
            else if (tagIDforwritingS3B0 == "2005630000000000000003B00")
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                         "2005630000000000000003B00", m_WriteParams,
m_AntennaInfoForm.getInfo());
                    tagIDforwaitongnextstep = "2005630000000000000003B01";
                }
                catch (Exception)
                {
                    MessageBox.Show("Could Not Write On Part");
                }
            }
            else if (tagIDforwritingS3B0 == "2005630000000000000004B00")
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                          "20056300000000000004B00", m_WriteParams,
m_AntennaInfoForm.getInfo());
                    tagIDforwaitongnextstep = "20056300000000000004B01";
                }
                catch (Exception)
                {
                    MessageBox.Show("Could Not Write On Part");
                }
            }
            else if (tagIDforwritingS3B0 == "20056300000000000005B00")
            {
                try
                {
                    m_ReaderAPI.Actions.TagAccess.WriteWait(
                           "20056300000000000005B00", m_WriteParams,
m_AntennaInfoForm.getInfo());
                    tagIDforwaitongnextstep = "20056300000000000005B01";
                }
                catch (Exception)
                {
                    MessageBox.Show("Could Not Write On Part");
                }
            }

            if (m_ReaderAPI.Actions.TagAccess.OperationSequence.Length > 0)
            {


m_ReaderAPI.Actions.TagAccess.OperationSequence.PerformSequence(m_AccessFilter
Form.getFilter(),
                    m_TriggerForm.getTriggerInfo(),
m_AntennaInfoForm.getInfo());
            }else{
                m_ReaderAPI.Actions.Inventory.Perform(
                    m_PostFilterForm.getFilter(),
                    m_TriggerForm.getTriggerInfo(),
                    m_AntennaInfoForm.getInfo());
            }
            for (int i = 0; i < 28; i++)
            {
                if (S3B1BackgroundWorker.CancellationPending)
                {
                    e.Cancel = true;
                    return;
                }
                Thread.Sleep(214);
                (sender as BackgroundWorker).ReportProgress((int)(100 / 28) *
i, null);
            }
            if (Station1.Station_State == "Free")
            {
                Station1.Waiting(tagIDforwaitongnextstep);
                if (Conveyor_Status == "START")
                {
                    Conveyor_Status = "STOP";
                    LPTPort = Conveyor1.Stop(LPTPort);
                    Robot3.PutOnCFromS3();
                    Thread.Sleep(3000);
```

```
    }
    else
    {
        while (true)
        {
            if (Conveyor_Status == "START")
            {
                Conveyor_Status = "STOP";
                LPTPort = Conveyor1.Stop(LPTPort);
                Robot3.PutOnCFromS3();
                Thread.Sleep(3000);
                break;
            }
        }
    }
    LPTPort = Conveyor1.Start(LPTPort);
    Conveyor_Status = "START";
    LPTPort = Station3.Free3(LPTPort);
}
else if (Station2.Station_State == "Free")
{
    Station2.Waiting(tagIDforwaitongnextstep);
    if (Conveyor_Status == "START")
    {
        Conveyor_Status = "STOP";
        LPTPort = Conveyor1.Stop(LPTPort);
        Robot3.PutOnCFromS3();
        Thread.Sleep(3000);
    }
    else
    {
        while (true)
        {
            if (Conveyor_Status == "START")
            {
                Conveyor_Status = "STOP";
                LPTPort = Conveyor1.Stop(LPTPort);
                Robot3.PutOnCFromS3();
                Thread.Sleep(3000);
                break;
            }
        }
    }
    LPTPort = Conveyor1.Start(LPTPort);
    Conveyor_Status = "START";
    LPTPort = Station3.Free3(LPTPort);
}
else
{
    if (BufferVar == "empty")
    {
        Robot3.PutOnBufferS3();
        BufferRed();
        BufferVar = "full";
        Thread.Sleep(3000
        LPTPort = Station3.Free3(LPTPort);
        BufferBackgroundWorker.RunWorkerAsync();
    }
    else
    {
        while (true)
        {
```

```
                    if (Station1.Station_State == "Free")
                    {
                        Station1.Waiting(tagIDforwaitongnextstep);
                        if (Conveyor_Status == "START")
                        {
                            Conveyor_Status = "STOP";
                            LPTPort = Conveyor1.Stop(LPTPort);
                            Robot3.PutOnCFromS3();
                            Thread.Sleep(3000);
                            //BufferGreen();
                            break;
                        }
                        else
                        {
                            while (true)
                            {
                                if (Conveyor_Status == "START")
                                {
                                    Conveyor_Status = "STOP";
                                    LPTPort = Conveyor1.Stop(LPTPort);
                                    Robot3.PutOnCFromS3();
                                    Thread.Sleep(3000);
                                    //BufferGreen();
                                    break;
                                }
                            }
                        }
                        LPTPort = Conveyor1.Start(LPTPort);
                        Conveyor_Status = "START";
                    }
                    if (Station2.Station_State == "Free")
                    {
                        Station2.Waiting(tagIDforwaitongnextstep);
                        if (Conveyor_Status == "START")
                        {
                            Conveyor_Status = "STOP";
                            LPTPort = Conveyor1.Stop(LPTPort);
                            Robot3.PutOnCFromS3();
                            Thread.Sleep(3000);
                            //BufferGreen();
                            break;
                        }else{
                            while (true)
                            {
                                if (Conveyor_Status == "START")
                                {
                                    Conveyor_Status = "STOP";
                                    LPTPort = Conveyor1.Stop(LPTPort);
                                    Robot3.PutOnCFromS3();
                                    Thread.Sleep(3000);
                                    //BufferGreen();
                                    break;
                                }
                            }
                        }
                        LPTPort = Conveyor1.Start(LPTPort);
                        Conveyor_Status = "START";
                    }
                }
            }
        }
    }
```

Appendix C: Code for Buffer Station

```
private void BufferBackgroundWorker_DoWork(object sender, DoWorkEventArgs e)
        {
            while (true)
            {
                if (BufferBackgroundWorker.CancellationPending)
                {
                    e.Cancel = true;
                    return;
                }
                if (Station1.Station_State == "Free")
                {
                    if (BufferBackgroundWorker.CancellationPending)
                    {
                        e.Cancel = true;
                        return;
                    }
                    Station1.Waiting(tagIDforwaitongnextstep);
                    if (Conveyor_Status == "START")
                    {
                        Conveyor_Status = "STOP";
                        LPTPort = Conveyor1.Stop(LPTPort);
                        Robot3.PutOnCFromBufferS3();
                        Thread.Sleep(3000);
                        break;
                    }
                    else
                    {
                        if (BufferBackgroundWorker.CancellationPending)
                        {
                            e.Cancel = true;
                            return;
                        }
                        while (true)
                        {
                            if (BufferBackgroundWorker.CancellationPending)
                            {
                                e.Cancel = true;
                                return;
                            }
                            if (Conveyor_Status == "START")
                            {
                                Conveyor_Status = "STOP";
                                LPTPort = Conveyor1.Stop(LPTPort);
                                Robot3.PutOnCFromBufferS3();
                                Thread.Sleep(3000);
                                break;
                            }
                        }
                    }
                    LPTPort = Conveyor1.Start(LPTPort);
                    Conveyor_Status = "START";
                }
                if (Station2.Station_State == "Free")
                {
                    if (BufferBackgroundWorker.CancellationPending)
                    {
                        e.Cancel = true;
                        return;
                    }
                    Station2.Waiting(tagIDforwaitongnextstep);
                    if (Conveyor_Status == "START")
                    {
                        Conveyor_Status = "STOP";
                        LPTPort = Conveyor1.Stop(LPTPort);
                        Robot3.PutOnCFromBufferS3();
                        Thread.Sleep(3000);
                        break;
                    }
                    else
                    {
                        if (BufferBackgroundWorker.CancellationPending)
                        {
                            e.Cancel = true;
                            return;
                        }
                        while (true)
                        {
                            if (BufferBackgroundWorker.CancellationPending)
                            {
                                e.Cancel = true;
                                return;
                            }
                            if (Conveyor_Status == "START")
                            {
                                Conveyor_Status = "STOP";
                                LPTPort = Conveyor1.Stop(LPTPort);
                                Robot3.PutOnCFromBufferS3();
                                Thread.Sleep(3000);
                                break;
                            }
                        }
                    }
                    LPTPort = Conveyor1.Start(LPTPort);
                    Conveyor_Status = "START";
                }
            }

        }
```

Appendix D: Code for Connecting to Reader

```
        private void connectBackgroundWorker_DoWork(object sender,
DoWorkEventArgs workEventArgs)
        {
            connectBackgroundWorker.ReportProgress(0, workEventArgs.Argument);

            if ((string)workEventArgs.Argument == "Connect To The Reader")
            {
                m_ReaderAPI = new RFIDReader(ReaderIP, ReaderPort, 0);

                try
                {
                    m_ReaderAPI.Connect();
                    m_IsConnected = true;
                    workEventArgs.Result = "Connect Succeed";

                }
                catch (OperationFailureException operationException)
                {
                    workEventArgs.Result = operationException.Result;
                }
                catch (Exception ex)
                {
                    workEventArgs.Result = ex.Message;
                }
            }
            else if ((string)workEventArgs.Argument == "Disconnect From The
Reader")
            {
                try
                {
                    m_ReaderAPI.Disconnect();
                    m_IsConnected = false;
                    workEventArgs.Result = "Disconnect Succeed";

                }
                catch (OperationFailureException ofe)
                {
                    workEventArgs.Result = ofe.Result;
                }
            }
        }

        private void connectBackgroundWorker_ProgressChanged(object sender,
            ProgressChangedEventArgs progressEventArgs)
        {
            m_ConnectionForm.connectionButton.Enabled = false;
            button1.Enabled = false;
        }

        private void connectBackgroundWorker_RunWorkerCompleted(object sender,
            RunWorkerCompletedEventArgs connectEventArgs)
        {
            if (button1.Text == "Connect To The Reader")
            {
                if (connectEventArgs.Result.ToString() == "Connect Succeed")
                {
                    /*
```

```
                     *  UI Updates
                     */
                    button1.Text = "Disconnect From The Reader";
                    m_ConnectionForm.hostname_TB.Enabled = false;
                    m_ConnectionForm.port_TB.Enabled = false;
                    m_ConnectionForm.Close();
                    this.readButton.Enabled = true;
                    this.readButton.Text = "Start System";
                    blockEraseToolStripMenuItem.Enabled =
m_ReaderAPI.ReaderCapabilities.IsBlockEraseSupported;
                    blockWriteToolStripMenuItem.Enabled =
m_ReaderAPI.ReaderCapabilities.IsBlockWriteSupported;

                    /*
                     *  Events Registration
                     */
                    m_ReaderAPI.Actions.PreFilters.DeleteAll();

                    m_ReaderAPI.Events.ReadNotify += new
Events.ReadNotifyHandler(Events_ReadNotify);
                    m_ReaderAPI.Events.AttachTagDataWithReadEvent = false;
                    m_ReaderAPI.Events.StatusNotify += new
Events.StatusNotifyHandler(Events_StatusNotify);
                    m_ReaderAPI.Events.NotifyGPIEvent = true;
                    m_ReaderAPI.Events.NotifyAntennaEvent = true;
                    m_ReaderAPI.Events.NotifyReaderDisconnectEvent = true;
                    m_ReaderAPI.Events.NotifyBufferFullEvent = true;
                    m_ReaderAPI.Events.NotifyBufferFullWarningEvent = true;
                    m_ReaderAPI.Events.NotifyAccessStartEvent = true;
                    m_ReaderAPI.Events.NotifyAccessStopEvent = true;
                    m_ReaderAPI.Events.NotifyInventoryStartEvent = true;
                    m_ReaderAPI.Events.NotifyInventoryStopEvent = true;
                    m_ReaderAPI.Events.NotifyReaderExceptionEvent = true;

                    //this.Text = "Connected to " + m_ConnectionForm.IpText;
                    this.connectionStatus.BackgroundImage =

global::Majid_RFID_Project.Properties.Resources.connected;
                    configureMenuItemsUponConnectDisconnect();
                    configureMenuItemsBasedOnCapabilities();

m_ReaderAPI.Actions.TagAccess.OperationSequence.DeleteAll();
                    if (memBank_CB.SelectedIndex >= 1)
                    {
                        TagAccess.Sequence.Operation op = new
TagAccess.Sequence.Operation();
                        op.AccessOperationCode =
ACCESS_OPERATION_CODE.ACCESS_OPERATION_READ;
                        op.ReadAccessParams.MemoryBank =
(MEMORY_BANK)1;//memBank_CB.SelectedIndex - 1;
                        op.ReadAccessParams.ByteCount = 0;
                        op.ReadAccessParams.ByteOffset =
m_ReadForm.m_ReadParams.ByteOffset;
                        op.ReadAccessParams.AccessPassword =
m_ReadForm.m_ReadParams.AccessPassword;

m_ReaderAPI.Actions.TagAccess.OperationSequence.Add(op);
                    }
                    Robot1.Reference1();
                    Thread.Sleep(500);
                    Robot2.Reference2();
                    Thread.Sleep(500);
```

```
                        Robot3.Reference3();
                    }
                }
                else if (button1.Text == "Disconnect From The Reader")
                {
                    if (connectEventArgs.Result.ToString() == "Disconnect 
Succeed")
                    {
                    }
                    //this.Text = "CS_RFID3_Host_Sample2";
                    this.connectionStatus.BackgroundImage = 

global::Majid_RFID_Project.Properties.Resources.disconnected;

                    button1.Text = "Connect To The Reader";
                    m_ConnectionForm.hostname_TB.Enabled = true;
                    m_ConnectionForm.port_TB.Enabled = true;
                    this.readButton.Enabled = false;
                    this.readButton.Text = "Start System";
                    configureMenuItemsUponConnectDisconnect();
                    m_IsConnected = false;

                }
                functionCallStatusLabel.Text = connectEventArgs.Result.ToString();
                //m_ConnectionForm.connectionButton.Enabled = true;
                button1.Enabled = true;

                updateGPIState();
            }
```