INTELLIGENT SINGLE WORKER UTILIZATION FOR COMPONENT RETRIEVAL FOR MULTIPLE STATIONARY ASSEMBLY LINES

By

ELIE MICHAEL NGANDU

Thesis submitted in fulfillment of the requirements for the degree:

MASTER TECHNOLOGIAE: ENGINEERING ELECTRICAL

in the

School of Electrical, Electronic and Computer Engineering

of the

Faculty of Engineering, Information and Communication Technology

at the

Central University of Technology, Free State

Supervisor:

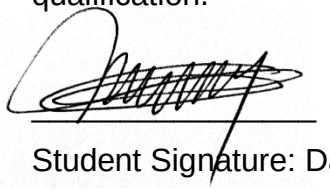
Dr. NJ LUWES

Bloemfontein

2019

DECLARATION OF INDEPENDENT WORK

I, **Elie Michael Ngandu** (Identity number: Student number:) do hereby declare that this research project which has been submitted to the Central University of Technology for the degree MASTER TECHNOLOGIAE: ENGINEERING ELECTRICAL, is my independent work; and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology; and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.



Student Signature: Date: 2019 – 08 – 20

ACKNOWLEDGEMENTS

The Author would like to acknowledge the following individuals and institute, whom without whom the completion of this dissertation would not have been possible:

- Firstly, to The God Almighty, because through Him all is possible, and He surely works in mysterious ways!
- Dr NJ Luwes, Prof. HJ Vermaak, the CUT, and RGEMS, for the provision of guidance, granting the opportunity to undertake the project, monitor assistance and the work experience gained during the duration of the project.
- To JA Niemann for his assistance and advice during the development of this project.
- To the CUT cyber junkyard 2012 team for their much-appreciated help in building a section of this project.
- And to various fellow research students within the RGEMS research group.

ABSTRACT

Since the late 90s, the manufacturing industries have considerably evolved. Industries are more frequently making use of automated systems to improve their productivity, reduce the assembling time, and to be more competitive. This has led to an ever-increasing necessity for the implementation of automated manufacturing system capable of effectively handling all the required manufacturing processes.

This study aims to develop an automated manufacturing system that will be easy to manage, maintain, and capable of efficiently using the resources to complete the required manufacturing tasks. This experimental system consists of two assembly lines, an AGV, and a storeroom. These parts have been developed as autonomous system that collaborate with one another over a network link.

The assembly lines use a programmed algorithm that help them share information about each task performed, negotiate on the usage of the autonomous guided vehicle (AGV) and storeroom, and backup each other's data. Each task performed by the assembly line is a time-based simulation of the assembling process of the four parts required from the storeroom. An AGV is used by the assembly line to collect parts from the storeroom. It follows a black line on a white surface connecting the storeroom to the assembly lines by using digital sensors. The storeroom uses an RFID system to store and retrieve parts according to the request placed by the assembly lines.

The final results obtained show that the project's main objectives to create a resource sharing algorithm in a manufacturing system with autonomous assembly lines, AGV, and storeroom, capable of negotiating and sharing information were successfully achieved.

TABLE OF CONTENTS

List of tables

List of figures

Acronyms and Abbreviations

AGV	Automated Guided Vehicle
ALB	Assembly Line Balancing
ASP	Assembly Sequence Planning
IEEE	Institute of Electrical and Electronics Engineers
IR	Infra-Red
LAN	Local Area Network
LED	Light Emitting Diode
MOACO	Multi-Object Ant Colony Optimization
OOP	Object Oriented Programming
PLC	Programmable Logic Controller
RFID	Radio Frequency Identification
SUALBSP	Setup Assembly Line Balancing and Scheduling Problem
TCP/IP	Transmission Control Protocol/Internet Protocol
TSALBP	Time and Space Assembly Line Balancing Problem
UDP	User Datagram Protocol

1 INTRODUCTION

Manufacturing and workshop practices have become important in the industrial environment to produce products for the service of humanity. It is the backbone of any industrialized nation [1].

A manufacturing system is a collection of integrated systems whose function is to perform one or more processing and assembly operations on a starting raw material, part, or set of parts [2].

In the late nineteenth and early twentieth century, the rapid indutrialisation of countries led to a need to improve the productiveness of factories. Frederick Winslow Taylor was a precursor in increasing workers' efficiency [3]. He believed that work could be managed by using scientific methods. He planned methods for reducing the movements required to complete the tasks, thereby increasing worker output. Taylor's principles were ultimately used to create assembly lines, in which workers were placed along a mechanized line. The product was put on the line, and each worker completed a single and specialized task over and over again. Because the product came to the workers, their work required very little movement. This is called the specialization of labor. The man who made the assembly line famous was Henry Ford. The assembly line increased efficiency but also minimized variety and originality. Automobiles before this point bare the mark of expert craftsmen, and each was somewhat unique [4].

The manufacturing industries has significantly involved, and the ever increasing use of machineries and automation have led to a rising need for an autonomous manufacturing system that is capable of efficiently managing the tasks and resources at its disposal [5].

This research proposes the development of a resource sharing algorithm for a manufacturing system that consists of two independent assembly lines, an AGV, and a storeroom. The two separate assembly lines will be connected to a local area network (LAN) to communicate and share information about the task they perform. They will negociate about the usage and the sharing of the resources (the AGV and the Storeroom). The assembly lines will be refer as the assembly lines and the AGV as the worker in the rest of this document.

Thus, this research focuses on the development of an algorithm that will allow the assembly devices to negociate for and share resources.

The case study set up consists of a worker that moves from the storeroom to either one of the assembly lines to supply them with the parts needed for the predefined tasks that they need to perform. The system layout is as follow:

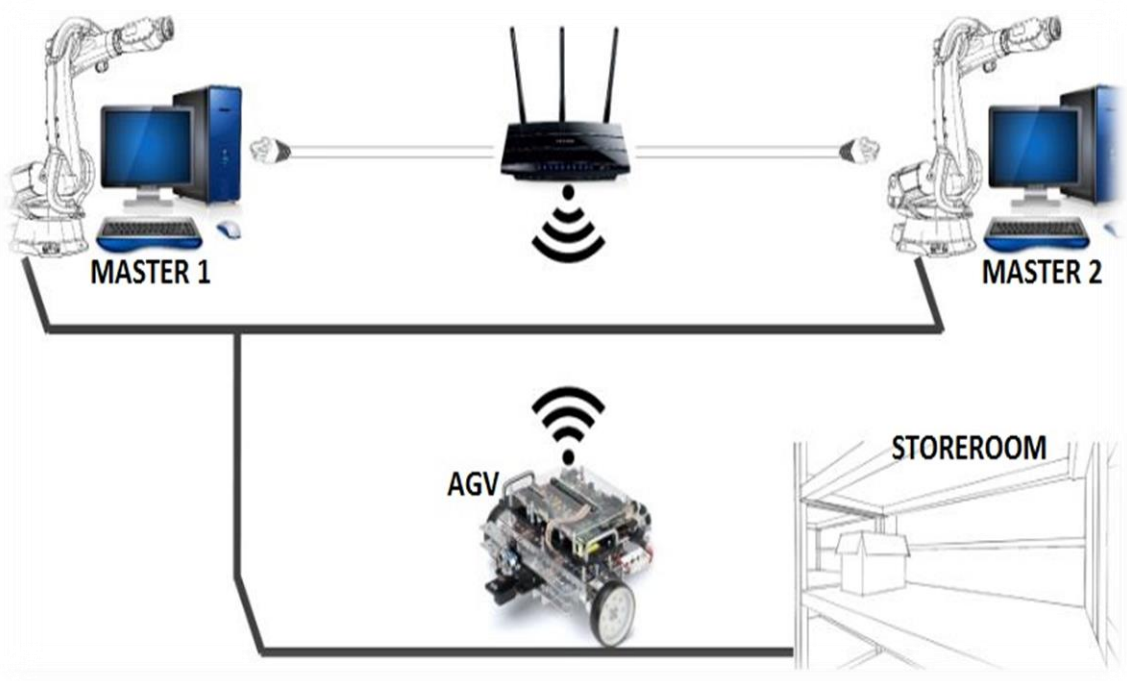


Figure 1.1 System layout

Figure 1.1 above shows that there are three layers to this system. This system will be composed of four different autonomous entities capable of communicating and collaborating with each other to achieve a set of goals preloaded to the system by the user at the initialization. The four different entities will be:

- Two assembly lines station
- An AGV worker
- The storeroom

Table 1.1 below is a detailed illustration of an assumption of all simulated tasks that the system will be expected to perform. In this illustration, there are four different jobs, each job has a level of priority, with level1 being the highest priority and level4 the lowest. Those levels of priorities will be used by the two assembly lines to determine which one of them gets to utilize the resources (AGV worker and Storeroom) first.

Each simulated job requires four parts to be completed, and each part has to be retrieved from the storeroom and assembled at the master's station.

Table 1.1 List of tasks to be performed

Tasks	Priority level	Components needed	Retrieval time in min.	Assembling time in min.	Total time in min.
JOB1	LEVEL1	PART2	2	10	12
		PART3	3	15	18
		PART3	3	15	18
		PART4	4	20	24
JOB1 COMPLETION TIME IN MINUTES					72
JOB2	LEVEL2	PART1	1	5	6
		PART2	2	10	12
		PART2	2	10	12
		PART3	3	15	18
JOB2 COMPLETION TIME IN MINUTES					48
JOB3	LEVEL3	PART1	1	5	6
		PART1	1	5	6
		PART2	2	10	12
		PART3	3	15	18
JOB3 COMPLETION TIME IN MINUTES					42
JOB4	LEVEL4	PART1	1	5	6
		PART1	1	5	6
		PART2	2	10	12
		PART2	2	10	12
JOB4 COMPLETION TIME IN MINUTES					36

Table 1.1 above will be used in the master's software application to determine their behaviours for cases where the two assembly lines start jobs simultaneously and negotiate on the utilization of the resources (worker & the storeroom). The master with the highest priority will use the worker first and send a message to the second assembly lines as soon as the worker is no longer required. Subcases will also be developed for which the worker can be released before the completion of the job. For example: if the first master is performing the highest priority job and for that job, the first part required is part 4 which takes four minutes to be collected from the storeroom and the twenty minutes to be assembled which make a total time of twenty-four minutes. And if the second master waiting for the worker is performing a job where the first three parts required are part 1 which takes one minute to be collected and five minutes to be assembled giving a total time of six minutes. Thus, the first master can release the worker after the collection of its first required part and then be busy for twenty minutes while the second master can use the worker for eighteen minutes and send it back to the first master with the highest priority job.

1.1 Problem statement

In a manufacturing system where two assembly lines share one AGV to collect part to complete a list of tasks, the system is faced with a problem where one assembly holds priority on the AGV most of the time leaving the other waiting for a long time to complete its task. In other words where manufacturing industries are always evolving. To be more competitive industries are more frequently making use of automated systems to improve their productivity and reduce the assembling time.

1.2 Research Goals and Objectives

1.2.1 Hypothesis

A time-based priority algorithm could be implemented on the two assembly lines to efficiently organize the negotiations and the sharing of the AGV for part collection from a storeroom

1.2.2 Specific Objectives

The aim of this study are:

- to develop a manufacturing system that will consist of two autonomous assembly lines, an AGV and a storeroom.
- To implement communication protocols between the assembly lines, AGV, and the storeroom for information sharing.
- To develop an algorithm that allows the assembly lines to efficiently negotiate for the utilization of the AGV and storeroom.

1.3 Layout of Dissertation

Chapter 1: an introductory chapter, which contains the problem statement, hypothesis and the objectives of the project.

Chapter 2: a literature study done by the author, to acquire preliminary knowledge in the field of study.

Chapter 3: a section containing a description of the method and steps followed to complete the project.

Chapter 4: a section dedicated to the tests that were done, how they were done, and the results obtained.

Chapter 5: A closing chapter to discuss test results as well as future work.

2 LITERATURE REVIEW

This chapter encloses a literature study done to acquire preliminary knowledge in the field of study. It highlights the different sections of the project and the selection of components, programming software, styles of programming selected to complete the research.

This research project a set for a small manufacturing system where a variation of tasks have to be performed by two assembly lines that are supplied with parts from a storeroom by an AGV.

This project proposes that the assembly lines (assembly lines), single AGV (worker) and storeroom be individual intelligent systems. These different intelligent systems would have the ability to make compromises and negotiate on which one of them will get to utilize the worker efficiently and as timely as possible, improving assembly time and throughput, thus improving the manufacturing capability of the plant. The project consists thus of assembly lines (assembly lines), single AGV (worker) and storeroom.

Therefore, this chapter is structured in the following subsection:

- Assembly line: this section of the chapter covers the research done on assembly line and the software used to program and simulate its functions
- The AGV: different types of AGVs are proposed in this section, and the justification of the selected type of AGV and sensor are also discussed.
- The storeroom: in this part of the chapter, the author highlight some of the components that are used to make an effective automated storage and retrieval system
- The communication: this is an essential part of research, and the author highlights the importance of reliable communication link between the different section of this research project

2.1 Assembly lines

An assembly line is a manufacturing method which is a crucial part of industry. It can be regarded as “a manufacturing process that is used in the mass production of standardized products, such as automobiles”. An assembly line is often composed of numerous stations organized in a serial manner and connected by a material handling system, such as a conveyor belt and AGVs. Parts of the product are sequentially launched down the line and are moved from station to station by the material handling system. Each station is allocated an identical time, known as the cycle time, to complete one or more tasks on each product.[6]

2.1.1 Research done on assembly line optimization

This section highlights different types of researches that have been done to optimize the operation of assembly lines. These researches are mainly focused on solving Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) that often occur during the production and development on manufacturing goods. Researcher in the following articles use soft computing approaches to solve ALP and ALB.

- Özcan Mutlu, Olcay Polat, and Aliye Ayca Supciller developed an Iterative Genetic Algorithm that is aimed at solving the assembly line worker assignment and balancing problem. They have adopted modified bisection search, genetic algorithm and iterated local search to obtain search diversity and efficiency [7].
- Abdolmajid Yolmeh and Farhad Kianfar proposed a hybrid genetic algorithm to solve the setup assembly line balancing and scheduling problem (SUALBSP). A dynamic programming procedure is used to hybridize the algorithm that determine the assignment of task to assembly station [8].
- Abdolreza Roshani, Parviz Fattahi, Abdolhassan Roshani, Mohsen Salehi and Arezoo Roshani addressed the cost-oriented two-sided assembly line balancing problem. They have used a mix integer programming with a heuristic algorithm based on simulated annealing approached to optimally solve this problem on a medium and large-scale [9].
- Juan Rada-Vilela, Manuel Chicab, Óscar Cordón and Sergio Damasb researched the Time and Space Assembly Line Balancing problem (TSALBP). They have compared the performance of height different Multi-Objective Ant Colony Optimization (MOACO) algorithms on ten well-known problem illustrations to resolve such an intricate problem [10].

Many more researches have been done on assembly line optimization and balancing. Therefore, the assembly lines in this project will be autonomous and computer based where all the tasks and processes will be time-based simulation executed by the programming software researched in section 2.1.1 of this chapter.

2.1.2 Computer programming software

Computer programming is simply defined as the process of writing instruction that a computer can execute [11]. The instructions are usually written using software called programming or coding language. The Figure 2.1 below shows various types of coding languages and their ranking according to the IEEE programming spectrums 2014 ranking.

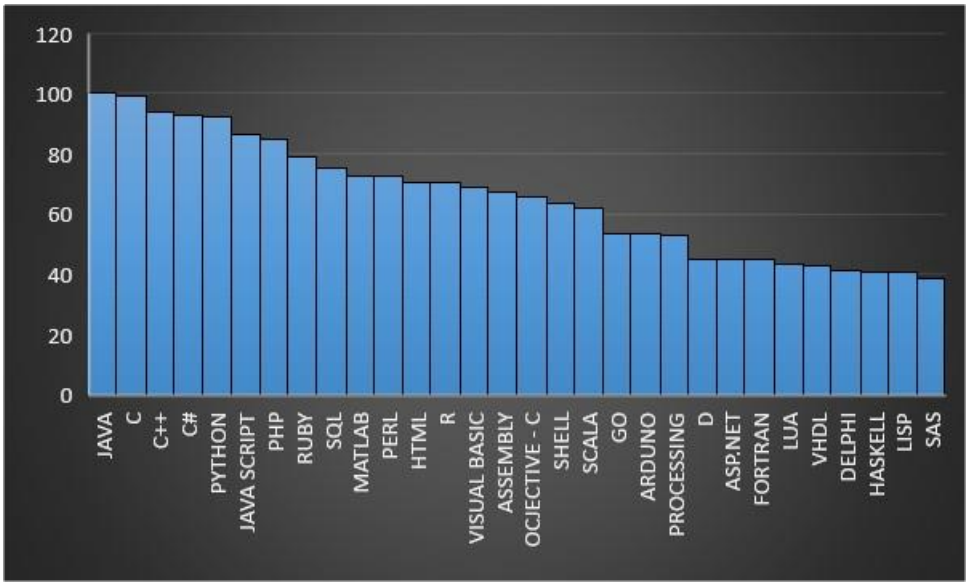


Figure 2.1 Various coding languages [12]

Out of the coding languages shown above in Figure 2.1 above, the author will only consider and compare the main programming languages that are currently being used in the area of computer application development. Those coding languages include C sharp (C#), Java, and Python

2.1.2.1 C Sharp (C#)

C# is a simple, modern, object-oriented, and type-safe programming language that combines the high efficiency of fast application development languages with the raw power of C and C++ [13]. The language provides support for software engineering principles like strong type examination, array bounds testing, detection of attempts to utilise uninitialized variables, and automatic garbage collection. Software robustness, durability, and programmer productivity are important. It is intended for developing software components suitable for deployment in distributed environments [13].

C# is envisioned to be appropriate for writing applications for both hosted and embedded systems, ranging from the very large that use refined operating systems, down to the very small having dedicated functions [13].

Although C# applications are envisioned to be efficient on memory and processing power necessities, the language was not intended to compete directly on performance and size with C or assembly language.

2.1.2.2 Java

The Java coding language is a general-purpose, concurrent, class-based, object-oriented language. It is intended to be simple enough that programmers can reach

fluency in the language. It is related to C and C++ but is structured slightly differently, with some aspects of C and C++ omitted and a few notions from other languages included. [14].

The Java coding language is a relatively high-level language and strongly typed. It has compile time usually consists of converting programs into a machine-independent byte code representation. Its run-time activities include loading and linking of the classes required to execute a program, optional machine code generation and dynamic expansion of the program, and actual program execution[14].

2.1.2.3 Python

Python is an interpreted, interactive, object-oriented programming language. It provides high-level data structures such as list and associative arrays (called dictionaries), dynamic typing and dynamic binding, modules, classes, exceptions, automatic memory management, etc... It has a remarkably straightforward and elegant syntax and yet is a powerful and general purpose programming language. It can be run on practically any modern computer. A python program is compiled automatically by the interpreter into platform independent bytecode which is then executed [15].

Python is modular by nature. The Python distribution includes a diverse library of standard extensions (some written in Python, others in C or C++) for operations ranging from string manipulations and Perl-like regular expressions, to Graphical User Interface (GUI) generators and including web-related utilities, operating system services, debugging and profiling tools, etc. New extension modules can be created to extend the language with new or legacy code. There are a substantial number of extension modules that have been developed and are distributed by members of the Python user community [15, 16].

Table 2.1 Main coding languages comparison

Language	Object oriented programming	Multi-threading	Event driven	TCP socket programming
C#	✓	✓	✓	✓
Java	✓	✓	×	✓
Python	✓	✓	×	✓

The above table is a comparison that the author uses to select the best suited coding language for the development of this study. It compares the main attributes of a coding language that are often used during the software development process.

2.1.2.4 Programming style

2.1.2.4.1 Object oriented programming

Object-oriented programming (OOP) is a programming pattern that represents the concept of "objects" that have data fields (attributes that describe the object) and associated processes known as methods. Objects, which are general illustrations of classes, are used to interact with one another to design applications and computer programs [17].

It is a programming style that focuses on the use of objects to design and build applications. An object can be comprehended as the model of the concepts, processes, or things in the real world that are significant to the application [18]

2.2 Single automated guided vehicle (worker)

An AGV is a robot that uses markers or wires in the floor to navigate itself in a particular environment. It may also use vision, magnets, or lasers for navigation depending on its application, refer to Figure 2.2. They are frequently employed in industrial applications to move materials from one place to another a manufacturing facility or a storeroom [19]. AGVs increase productivity and reduce costs by facilitating the automation of a manufacturing plant or storeroom. AGVs have a wide variety of applications in which they are mostly used to transport many different types of material including pallets, rolls, racks, carts, and containers [20] [21].

Figure 2.2 Various industrial AGVs [22]

Figure 2.2 above shows an example of the various forms of industrial AGVs.

2.2.1 Types of AGVs

The following are the existing types of AGVs:

2.2.1.1 AGVS Towing Vehicles

It is usually a towing vehicle that can also be called an automated guided tractor. It is often composed of flatbed trailers, pallet trucks, or custom trailers. This form of AGV is commonly considered as the first type of AGV introduced to the industrial world. It is generally used for large volumes (>450 kg) and long moving distances (>300m) [20].

2.2.1.2 AGVS Unit Load Carriers

This type of AGV is equipped with a powered or non-powered roller, chain or belt deck, or custom deck for carriage of individual unit load onboard the vehicle by Pallet truck, forklift truck, automatic loading/unloading equipment, etc. [20].

2.2.1.3 AGVS Pallet Trucks

These AVGs are generally used in distribution tasks with a load capacity ranging from 450-900 kg and usually has a maximum speed that is less than 60m/min. The pallet truck can be loaded onto the AGV either manually or automatically. This particular kind of AGV does not need a special device for loading except that the loads should be on a pallet. It is restricted to floor level loading and unloading with palletized load [20].

2.2.1.4 AGVS Forklift Trucks

This type of AGV is usually very costly and only used where complete automation is required. It has the ability to pick up and drop palletized load both at ground level and on stands. It can accurately place its fork according to load stands with different heights [20].

2.2.1.5 AGVS Light Load Transporters

This type of AGV is used to dispense packs/parts between storage and multiple workstations with a standard speed 30m/min. It is commonly used to handle light and small loads/parts over moderate distances with a load capacity that is less than 225 kg. They are usually used for manufacturing areas with restricted space because of its turning radius 0.61m making it ideal for these type of applications [20].

2.2.1.6 AGVS Assembly-Line Vehicles

This type of AGV is slightly similar to a light load transporter, and it is used for serial assembly processes. The AGV travels from one station to another, and subsequent assembly operations are performed. It offers flexibility for the manufacturing processes by dropping expenses and making it easy to install. However, a sophisticated computer control and extensive planning are essential to integrate the system [20].

2.2.2 AGV Guidance system

The aim of AGV guidance system is to keep the AGV on a predefined path. One of the main advantages of AGV is that it is simple to adjust by making use of the guidance system for altering the guide path at low cost compare to conveyors, chains, etc.

An additional benefit is that guide tracks are flexible to allow intersection on different paths because usually, a guide path does not obstruct another system. Various types of guidance systems can be selected by considering the type of AGV selected, its application, requirement, and environmental limitation [23]. The different kinds are as followed:

- **Wire-guided**: An antenna on the AGV follows an energized wire is rooted along the guide path [23].
- **Guide tape**: A tape is used for the guide path, and the AGV is fitted with the suitable guide sensor to follow the path of that tape. The tapes can either be magnetic or collared [23].

- **Inertia**l: The AGV uses sonar system or obstacle avoidance sensor to follow a guide path that is programmed on its microprocessor [23].
- **Infrared**: Reflectors are attached on the top of the AGV and infrared light transmitters are used to detect the position of the vehicle [23].
- **Laser**: Accurate positioning of the AGV can be obtained by using a laser beam to scan wall-mounted bar-coded reflectors [23].

- **Teaching type**: AGV learns the guide path by moving the required route and sends the information about every step taken to the host computer [23].

2.2.3 Line following sensors

These are the different types of sensors that could be used for a line following robot:

2.2.3.1 MGS1600 magnetic sensor

The MGS1600 is a sensor that is capable of detecting accurately and reporting the location of a magnetic field along its horizontal axis. This sensor is mostly used for line following robotic applications, using a magnetic tape to form a track guide on the floor. It uses innovative signal processing to measure its lateral distance from the centre of the track precisely, with millimetre resolution, resulting in nearly 160 points end to end. The information related to the position of the tape can be output in numerical format on the sensor's RS232 or USB ports [24]

Furthermore, this sensor will detect and report the presence of magnetic indicators that can be placed on the left or right side of the path. The sensor is equipped with four LEDs for easy monitoring and diagnostics [24].

2.2.3.2 Opto-electrical sensors

Optoelectronics is defined as “the study and application of electronic devices that source, detect and control light.” The opto-electrical sensor can be used to identify precisely light reflecting object from the non-reflecting object, example a Wight tape on a dark floor. Therefore, these sensors are also perfect for line following robotic applications where a light reflecting tape can be used the guide for the robot to follow.

The SOEG-RT-M12-PS-K-2L opto-electrical sensor by Festo is ideal for line following robotic applications. It has an operating voltage ranged from 10 to 30 V with a max output current of 200mA. It has a switching frequency that is up to 1kHz, and an operating temperature ranged between -25º and 55ºC. It is a retro-reflective sensor with a sensitivity range that can go up to 1.5m [25].

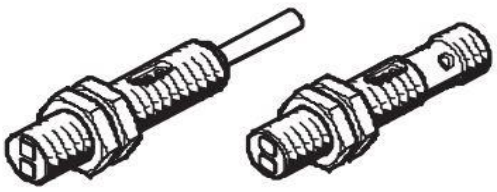


Figure 2.3 Festo SOEG-RT-M12-PS-K-2L opto-electrical sensor[24]

Figure 2.3 above is a drawing picture of the Festo SOEG-RT-M12-PS-K-2L opto-electrical sensor.

2.2.3.3 Infrared line following sensor

In this section is discussed, what is an Infrared sensor ad its working mechanism.

2.2.3.3.1 What is an IR sensor? [26]

An infrared sensor is used to detect infrared radiation that falls on it. Depending on the application, there are various kinds of IR sensors that can be used. The following are some example of different types of IR sensor and their use:

- Proximity sensors: Used in Edge Avoiding Robots and Touch Screen phones
- Contrast sensors: Used in Line Follower Robots
- Obstruction counters/sensors: Used for counting goods and in Burglar Alarms.

2.2.3.3.2 Working Mechanism

An IR sensor is essentially a device which is made of a pair of an IR LED and a photodiode. The IR LED produces IR radiation that is received by the photodiode and intensity of that reception dictates the output of the sensor. Different applications of the infrared sensor are developed depending on the way and amount of radiation from the IR LED reaches the photodiode.

Figure 2.4 below shows the IR LED placed right in front of the photodiode in such a way that nearly all the radiation emitted reaches the photodiode. This makes an invisible line of infrared radiation between the IR LED and the photodiode. This mechanism is used in burglar alarms and object counters.

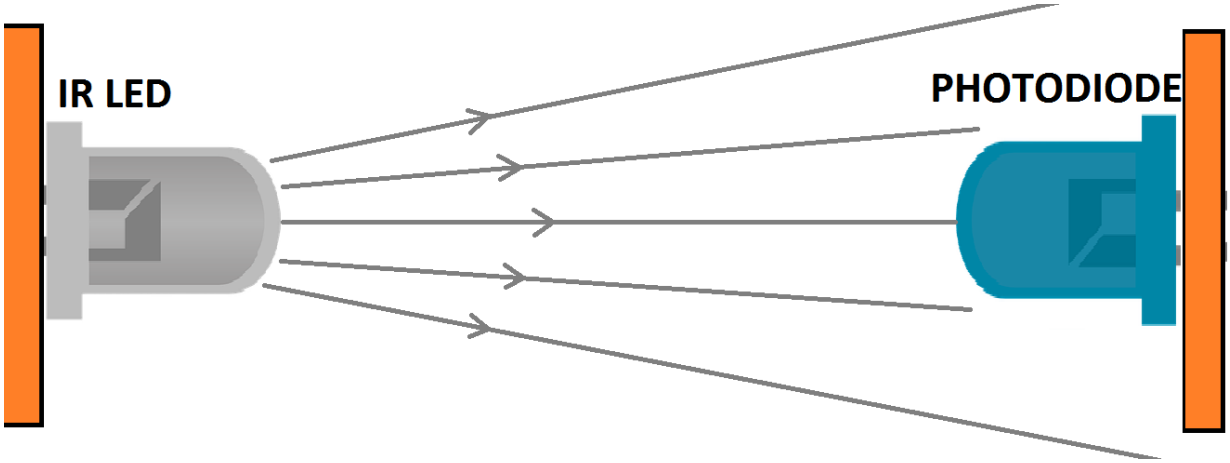


Figure 2.4 Direct incidence

Figure 2.5 below refers to a setting where the IR LED and the photodiode are placed side by side. All the IR radiation from the IR LED will be emitted away from the photodiode. This mechanism can be used to sense reflective objects or colours intensity. For instance, if a reflective object, (White or some other light colour), is placed in front of the IR LED and photodiode, then most of the radiation will get reflected from the object to the photodiode. If a non-reflective object is placed in front of the sensor, (Black or some other dark colour), then it will absorb most of the radiation, and none of it will be sensed by the photodiode. It is comparable to there being no surface (object) at all.

Figure 2.6 refers to circuit design used for IR sensors. They can be analog or digital:

- Analog sensor: they give an analog output voltage ranging from 0 to the voltage supply to the circuit (+5V for the circuit in Figure 2.6). That voltage is relative to the amount of radiation that the photodiode receives.
- Digital sensor: they give an output of 0 when there is no radiation reflected on the photodiode and 1 when the radiation is reflected onto the photodiode.

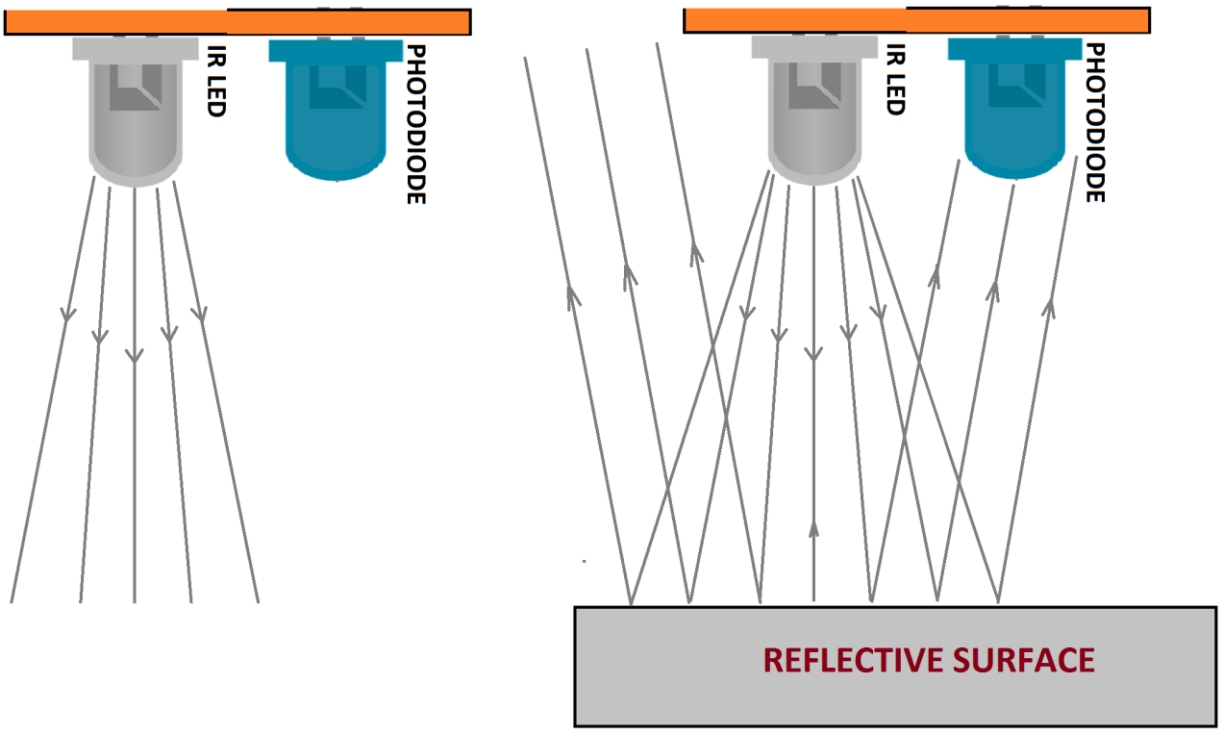


Figure 2.5 Indirect incidence

These different settings are shown in Figure 2.4 and Figure 2.5 are the basis for the many different types of IR sensor mention bellow:

- Proximity Sensors: uses Figure 2.5 setting. The amount radiation reflected back on the photodiode by an object is used to determine how close that object is from the sensor. The closer it is, the higher the intensity of the radiation on the photodiode will be.
- Line Follower Robots: uses Figure 2.5 setting. The IR LED generates IR radiation that in the standard state gets reflected to the sensor from the white surface around the black line. But, the instant that IR radiation falls on the black line, the radiation gets totally absorbed by the black colour, and there is no reflection of the IR radiation going back to the sensor module.
- Item Counter: uses Figure 2.4 setting. Every time an item obstructs the invisible line of IR radiation, an increment in the value of a stored variable in a computer/microcontroller which could be displayed by LEDs, Seven Segment Displays, and LCDs etc. These counters are often used to count products/parts on conveyor belts that are loaded/unloaded from large factories.
- Burglar Alarm: uses Figure 2.4 setting. Here, the IR LED is placed on one side of the door frame, and the photodiode on the other side, in such a way the IR

radiation emitted by the IR LED falls on the photodiode directly. As soon as a person obstructs the IR path, the alarm goes off.

Figure 2.6 below refers to the circuit diagram often used to build IR sensor. Depending on its application, the IR sensor can either be analog or digital. The Analog IR sensor gives out an output that ranges between zero and the maximum supplied voltage (5V for the circuit in Figure 2.6). The digital sensor provides an output of one when a significant amount of radiation falls on the photodiode and zero when there is no IR radiation falling the photodiode.

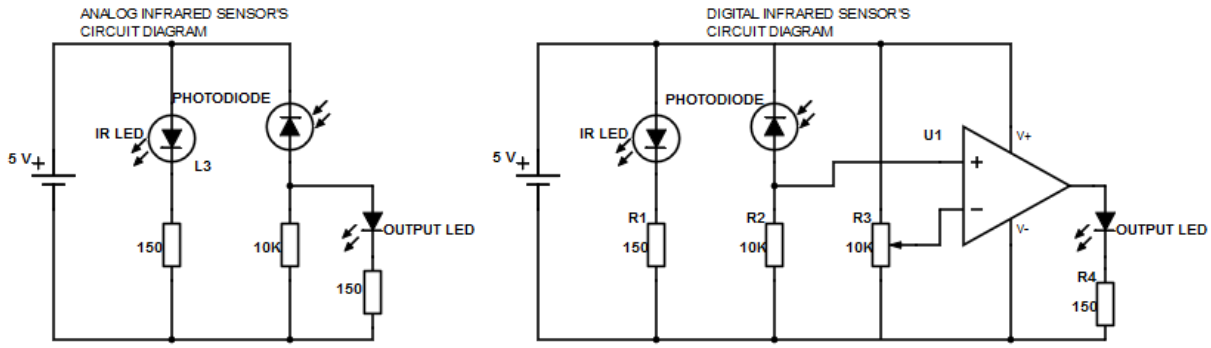


Figure 2.6 Analog and Digital circuit for IR sensors

The digital circuit in Figure 2.6 uses an op-amp as a comparator to change the analog voltage from the first part of the circuit to a digital signal. R3 is used the set the sensitivity of the IR sensor. The op-amp compares the voltage from the none-inverted (+) input that is connected to the analog voltage coming from the IR sensor to the voltage set on R3 connected to the inverted input of the op-amp (-). And if the positive voltage is greater than the negative voltage, the op-amp will give Vcc as the output and zero otherwise. These types of IR sensor when used for a line-follower robot, are usually required to be placed at least 1 cm from the line that needs to be followed.

2.2.4 LabVIEW

NI LabVIEW software is utilized for a wide diversity of applications and industries. It is a highly creative development environment for producing custom applications that interact with real-world data or signals in fields such as science and engineering.

The net result of using a tool such as LabVIEW is that advanced quality projects can be completed in less time with a smaller number of people involved.

Across different industries, the tools and components needed to succeed vary widely, and it can be an overwhelming task to find and use all these various items together. LabVIEW is exceptional because it makes this wide variety of tools accessible in a

single environment, making certain that compatibility is as simple as drawing wires between functions.

LabVIEW itself is a software development environment that comprises several components, a number of which are required for any type of test, measurement, or control application [27] [28].

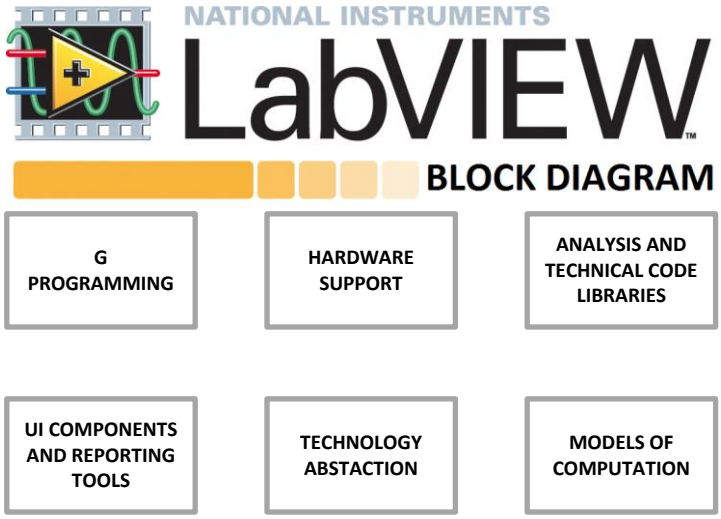


Figure 2.7 LabVIEW block diagram

Figure 2.7 above is the block diagram for LabVIEW, it shows the different blocks that make out the national instrument LabVIEW software.

2.3 Storeroom

The development of this study is set for a small manufacturing system where certain tasks have to be performed repeatively by two assembly lines that are supplied with parts from a storeroom by an AGV. Therefore, the storeroom provides all the necessary materials needed to maintain a smoothly operating manufacturing environment. Extra or lack of spares, unmanaged inventory and inefficient processes not only add cost, but also can negatively impact your production environment and operating budget. Therefore, a proper storage management system will significantly improve the effectiveness the manufacturing industry by improving the collection time and storage process. The storeroom will be an automated storage management system. In order to complete this section of the project, there are different aspects there are essential its successful completion.

1. The first aspect to look at is automation. A brief definition of what it is and what will be used to incorporate automation into this project.
2. The second aspect is the identification process using radio frequency identification
3. The third aspect is the motion control by making use of stepper motor for their precision to replace forklift movements.

2.3.1 Automation

Automation is the usage of information technologies, control systems and machines to enhance productivity in the production of goods and delivery of services. The reason for applying automation is to increase production, and/or quality beyond what is presently possible with human labour levels so as to realize predictable quality levels [2]. In the scope of industrialization, automation is a step beyond mechanization. While mechanization offers human operators with machinery to support them with the muscular requirements of work, automation significantly decreases the need for human physical and mental requirements but with increasing load capacity, speed, and repeatability. Automation plays an ever more important role in the world economy and in daily experience [2].

2.3.1.1 Programmable Logic Controller (PLC)

A Programmable Logic Controller, PLC or Programmable Controller is a digital computer used for automation of electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or light fixtures [29, 30]. Many

industries and machines are more and more making use of PLCs. Unlike general-purpose computers, it is intended for multiple inputs and output arrangements, extended temperature ranges, resistance to electrical noise, and can withstand vibration and impact [31].

Programmable Logic Controllers were first implemented as a replacement for automatic control systems that often used tens and hundreds (maybe even thousands) of hard wired relays, motor driven cam timers and rotary sequencers [31].

Figure 2.8 various types of PLCs

Figure 2.8 above shows various types of industrial PLCs.

2.3.1.2 Operation of a PLC

A Programmable Logic Controller is a device that is programmed to perform a series of events. These events are triggered by inputs received at the programmable logic controller through delayed actions such as time delays or counted occurrences. Once an event is triggered, it actuates in the outside world by switching on or off electronic control gear or the physical actuation of devices. A PLC will continually loop through its user defined program waiting for inputs and giving outputs at a specific programmed times [32].

2.3.1.3 Programming

PLC programs are usually written in a special software on a computer, then downloaded over a network to the PLC. The program is saved in the PLC non-volatile flash memory. Often, a single PLC can be programmed to replace thousands of relays.

PLCs are often programmed using standards-based programming languages. A graphical programming notation called Sequential Function Charts is available on certain programmable controllers. Originally most PLCs use Ladder Logic Diagram Programming, a model which emulated electromechanical control panel devices (such as the contact and coils of relays) which PLCs replaced. This model remains common today.

2.3.2 Radio frequency identification

Radio Frequency Identification (RFID) is a technology that is commonly used identify objects or people automatically. RFID systems consist of tags containing electronically stored information and readers that use radio frequencies to transfer identification data wirelessly from a tag to a reader on request of a querying reader [19]. An RFID tag is a small device containing of a microchip with limited functionality, data storage, and an antenna to wirelessly communicate with the readers. Depending on the powering technique, RFID tags can be passive, active or semi-active. In general passive tags have no on-board power and are powered from the signal of the interrogating reader. Active tags contain batteries for their transmission. Each RFID tag contains a unique identifier to serve as object identity so that this identity can be used as a link to relate information about the corresponding object. [33].

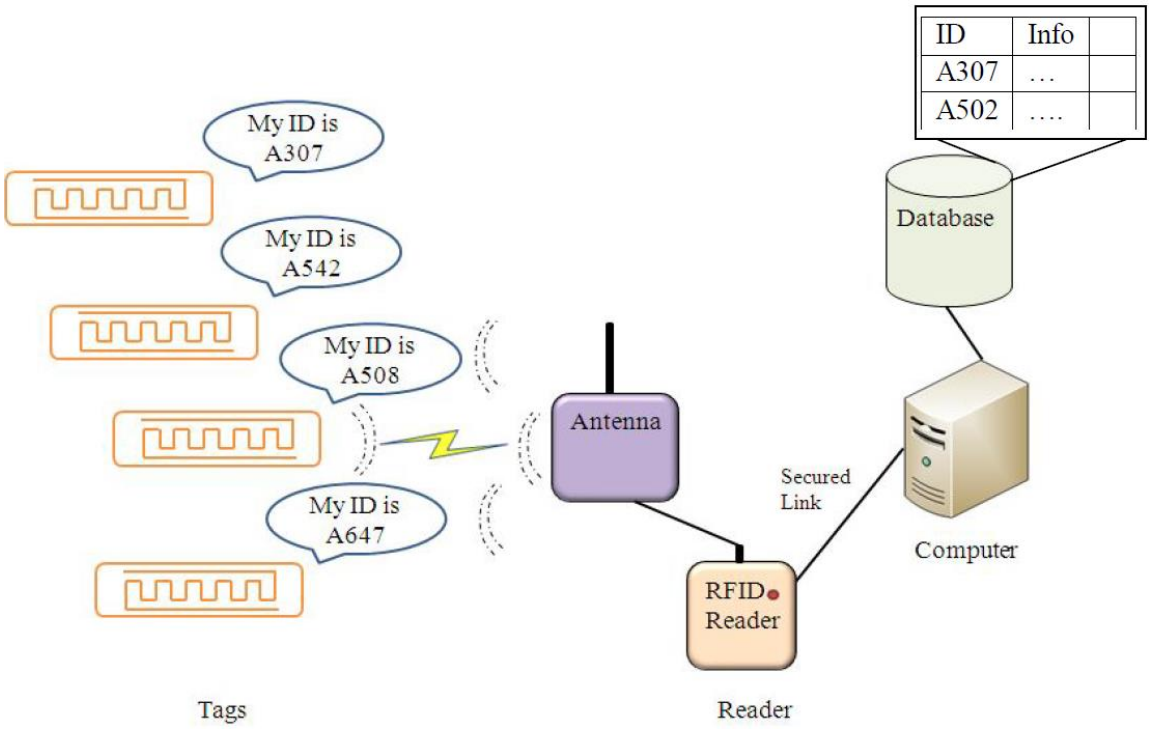


Figure 2.9 Typical RFID system's layout[33]

Figure 2.9 above is an illustration of a typical RFID system's layout. It shows how the RFID tags are activated by the antenna, read by the reader, and finally stored in a database by making used of a secured link between the computer and the RFID reader.

2.3.2.1 Uses

RFID systems are used for the following:

- General transport (logistics), tracking a package, parcel; replacing barcodes
- Tracking vehicles for road toll
- Many countries have started using RFID chips in passports
- Identifying animals; used for tracking pets, but also for research, for example on turtles.
- Keys for vehicles. The vehicle key has an RFID tag inside; only the key with the right RFID tag can start the vehicle (this makes copying vehicle keys harder). Also used for locking/unlocking vehicles from a distance.
- Contactless identity cards, for example to regulate entry into certain areas; also used for ticketing, or public transport

2.3.3 Step Motors

The stepper motor is defined as an electromechanical device which actuates a train of step movememnts of shaft in response to a train of input pulses. The step movement may be angular or linear [34]. With the assistance of a stepper motor controller, step motors convert electrical energy into accurate mechanical motion. It rotates in a specific incremental distance per each step. The number of steps that are executed controls the degree of rotation of the motor's shaft. This characteristic makes step motors exceptional for positioning applications [35].

The stepper motor controller can very accurately control how far and how fast the stepper motor will rotate. The number of steps the motor executes is equal to the number of pulse commands it is given by the controller. A stepper will rotate a distance and at a rate that is proportional to the number and frequency of its pulse commands [35].

2.3.3.1 Basic Step Motor System

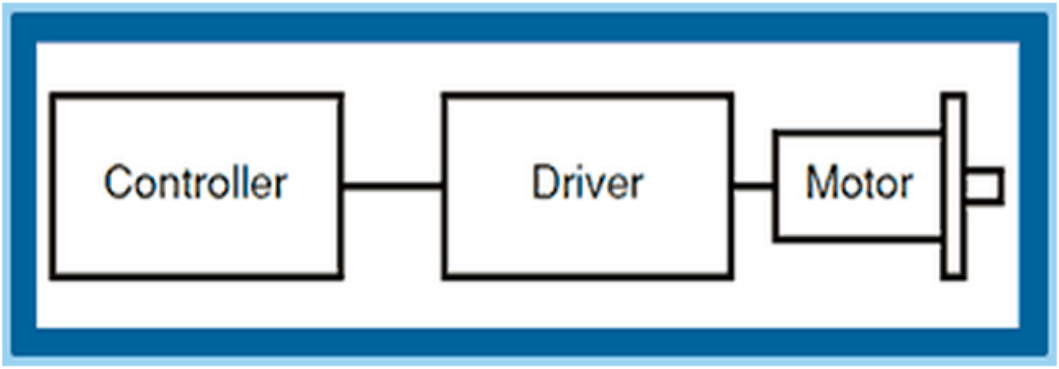


Figure 2.10 Basic step motor system [35]

Figure 2.10 is a diagram that shows a classic step motor based system. The stepper motor controller, step motor driver and motor must all be present in one form or another. Each component's performance will have an effect on the others [35].

2.3.3.2 Breakdown of Step Motor Benefits:

The stepper motor has the following benefits:

- Accuracy & Repeatability – Ability to position accurately.
- Responsiveness & Quick Acceleration – Step motors have low rotor inertia, allowing them to get up to speed quickly. This makes step motors an excellent choice for short, quick moves.
- Excellent torque for their size – Step motor has the highest torque per cubic inch of any motor.
- Positioning Stability – Unlike other types of motors, step motors can be held completely motionless in their stopped position.
- Cost and Reliability – Step motor technology is reliable and proven. It is the most cost effective method of precision position control.

2.4 Communication using Transmission Control Protocol/Internet Protocol (TCP/IP) with socket

A computer network is made up of machines interconnected by communication channels. These machines are called hosts and routers. The Hosts are computers running applications that are dependent on the network such as the Web browser [36]. A communication channel is a way to transmit sequences of bytes from one host to another; it could be a broadcast technology like Ethernet, a dial-up modem connection, or something more sophisticated [37].

Information transmitted over the network is a sequence of bytes that are constructed and interpreted by programs. These byte sequences are usually named packets. The packet contains control information that the network uses to do its job and occasionally also includes user data. An illustration will be information about the packet's destination. Routers use such control information to figure out how to forward each packet [37].

Protocol can be considered as an agreement about how packets are traded by communicating programs and what they mean. It tells in what way packets are organised—for instance, where the destination information is situated in the packet and how big it is—as well as how the information is to be interpreted [37].

The Implementation of a useful network necessitates that a lot of diverse problems be solved. To keep things manageable and modular, different protocols are designed to address different sets of problems. TCP/IP is one of such collection of solutions, sometimes called a protocol suite. The main protocols in the TCP/IP family are the Internet Protocol (IP), the Transmission Control Protocol (TCP), and the User Datagram Protocol (UDP) [37].

Transmission Control Protocol/Internet Protocol (TCP/IP) is the rudimentary communication protocol or language of the Internet. It can similarly be used as a communications protocol in a private network (either an intranet or an extranet). A copy of the TCP/IP program is provided to each computer when it set up with direct access to the internet. That program is used by each computer on the internet to send or receive information [38].

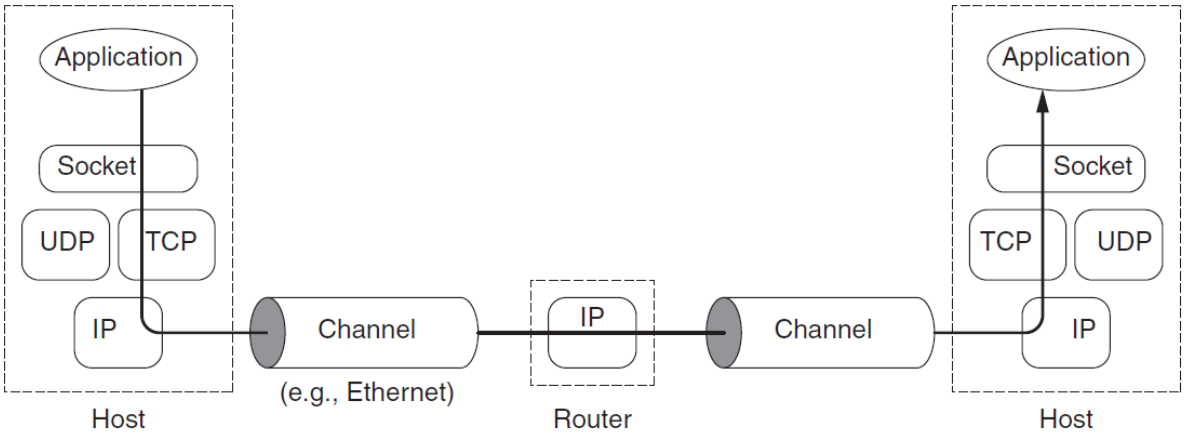


Figure 2.11 TCP/IP network.

Figure 2.11 illustrates the interactions between the applications, protocols, and the sockets API in the hosts and routers, along with the flow of data from one application (using TCP) to another[37].

A socket can be an abstraction over which an application can receive and send data. It allows the application to "connect" to a network and communicate with other applications that are also connected in to the same network. Information written to the socket by an application on one host can be read by an application on a different host, and vice versa[37].

2.4.1 Basic LabVIEW's TCP/IP Communication

TCP/IP communication in LabVIEW offers a simple user interface that conceals the complexities of guaranteeing consistent network communications. LabVIEW TCP/IP functions are located on the ***Functions » Communication » TCP palette***. The communication process usually involves opening the connection, reading and writing the information, and closing the connection [39].

Most communication always uses the client as a processor that initiates the connection to the server. With TCP/IP communication/connections, a computer can be able to function either as the server or the client. Figure 2.12 diagram below is a simple example of a client application that starts a connection to a remote server with TCP Open Connection. The server listens for remote connections and responds appropriately.

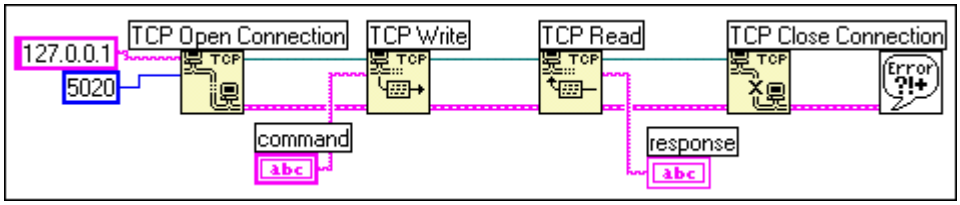


Figure 2.12 TCP Client Example with LabVIEW

LabVIEW allows users to develop a custom application for TCP/IP with client and server. This application can also be established with an access control server that uses the remote address output value of the TCP listen VI to determine if the remote client has permission to access the server.

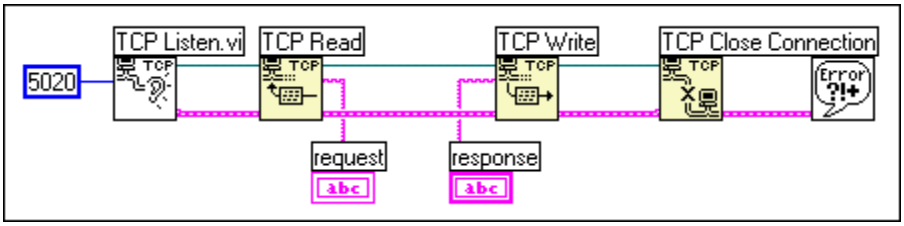


Figure 2.13 TCP Server example with LabVIEW

The following are the most important labview function blocks use for TCP communication.

- TCP Open Connection
- TCP read
- TCP write
- TCP close connection function

Detailed explanations, illustrations, and example of the above function blocks can be found in the NI LabVIEW user manual [40].

2.5 Chapter conclusion

This chapter covered the study and research for the potential hardware, software, and components that could be utilized in the implementation of this project. In the next chapter, the author will use some of these materials to develop the project proposed in this document. It will show how the hardware and components that are used and connected to the software selected. It also shows how the software are used to monitor and control the system.

3 METHODOLOGY

In this chapter, the author will cover and describe the method and process followed to build and program the different sections of this research project. Each of these subsections will discuss their contribution to the main objective of the creation of a resource sharing algorithm.

Referring to Figure 1.1, the system is divided into three separate subsections, which include:

- The master: in this section, the author illustrates the process that was followed to program the different functionalities of the assembly lines. It also contains the communication procedure used to share commands and information between the assembly lines and the resources.
- The worker: this section focuses on the method used to build and program an autonomous and accurate line follower robot. It also shows the different steps used to establish and maintain a proper communication line between the AGV and the assembly lines.
- The storeroom: this part of the chapter emphasizes on the development of an autonomous storage system capable of storing and retrieve parts according to their properties and specification.

3.1 System's manager operation (or the system's user)

Although this system is made of autonomous parts, it still needs a system's manager to get it started. The system's manager role is as followed:

- Write the list of tasks to be perform by the assembly lines

  The user writes a specific list of tasks that he wants the assembly lines to perform. Each one of the list has to be saved as text file with the following name ***"list jobs.txt".*** The task list only contains the name of the tasks without any other description. When the list is loaded to the assembly lines, the assembly lines will use these names to collect all the need information about each task from a task detailed Database that they contain.

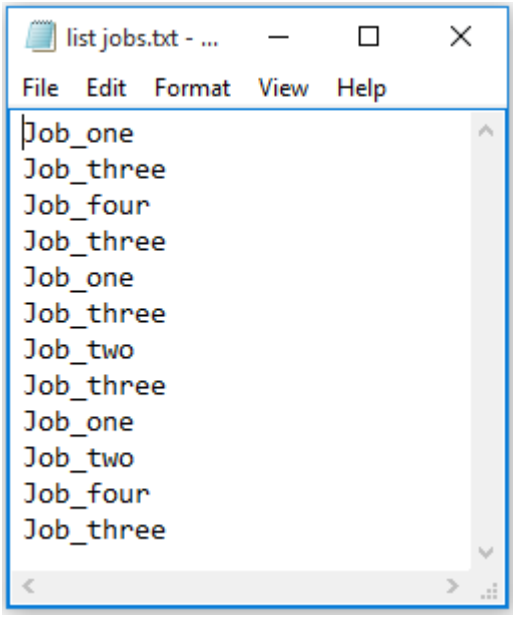


Figure 3.1 Task list

Figure 3.1 above is an example of the list of tasks written in Microsoft Notepad and it will be loaded into the assembly lines in the next step.

- Load that list onto the assembly lines

  After writing the task list, the user has to load it into a specific directory on the assembly lines computer which is ***"C:\Program Files\Assembly Line\Files"***. Once started, the assembly line will look for the task list in that directory to create a full list of all the tasks, parts and the time it takes to assemble them.
- Start the assembly lines
- Start the communication server

  Each assembly line can run the server's communication application, the system's manager has to select and start the server in one of the two assembly line to start the communication process.
- Connect all client to the server

  When the server is started, the system's manager has to make sure that all the assembly lines are connected to the correct communication server.
- Start the worker

  The system's manager turns on the AGV and places it at the start up position at the storeroom.
- Start the storeroom system

  The system's manager starts the storeroom system.
- Monitor the system's operations

The system's manager is a crucial part of this research because without it, none of the operations mentioned above will be possible. However, after the start of the system and every one of its components, it will run autonomously without the intervention of the system's manager.

3.2 The assembly line

Referring to Figure 1.1 in chapter one, it is shown that the assembly lines are computers connected to each other over a local area network. They share information about all the tasks and operation they perform over that network. The program and functions of the assembly lines were written using C#.net in Microsoft Visual Studio (refer to 2.1.2 Computer programming software in Chapter 2).

Each assembly line is programmed to follow these specific steps of its work cycle:

- Step1: read the preloaded list of jobs to be performed.
  At the initialisation of the assembly line, the system's manager has to load a list of jobs to be performed. The list contains the label of each task, its priority level, the parts needed for that job, the time required the collect each part from the storeroom, and the time necessary for the assembly line to assemble the job.
- Step2: calculate the total time needed to complete all the tasks on the list.
- Step3: share step1 and step2 over the network with the other assembly line.
  At this step the assembly lines share all the information regarding the jobs to be performed and those that have been performed already with the other assembly line. This will serve as a backup in case of a breakdown or maintenance of the assembly lines.
- Step4: start the first/next job on the list.
- Step5: compare job's level of priority with the other assembly line
  If the job starting has the highest priority, the assembly line will go to step 6. Otherwise it will have to wait for the other assembly line to release the worker then execute step 6.
- Step6: order the parts need from the worker
  The assembly line sends a message to the work and the storeroom over the LAN about the part that is required. Each part that is delivered by the worker is scanned by the RFID scanner. Once all parts have been collected the assembly lines broadcast a massage to let the other assembly line's station know that the worker will no longer be needed for a specific amount of time.
- Step7: assemble and complete the job.

- Step 8: repeat process from step 4.

The flowchart in Figure 3.2 below illustrates the above-mentioned steps.

The assembly line's development is divided into 2 phases that complement each other. The first phase is to establish communication between the assembly lines and the other devices used in the project. This is discussed in detail in section 3.2.1 below. The second phase is the implementation of the assembly line's functions and operations. It will be discussed in section 3.2.2 below.

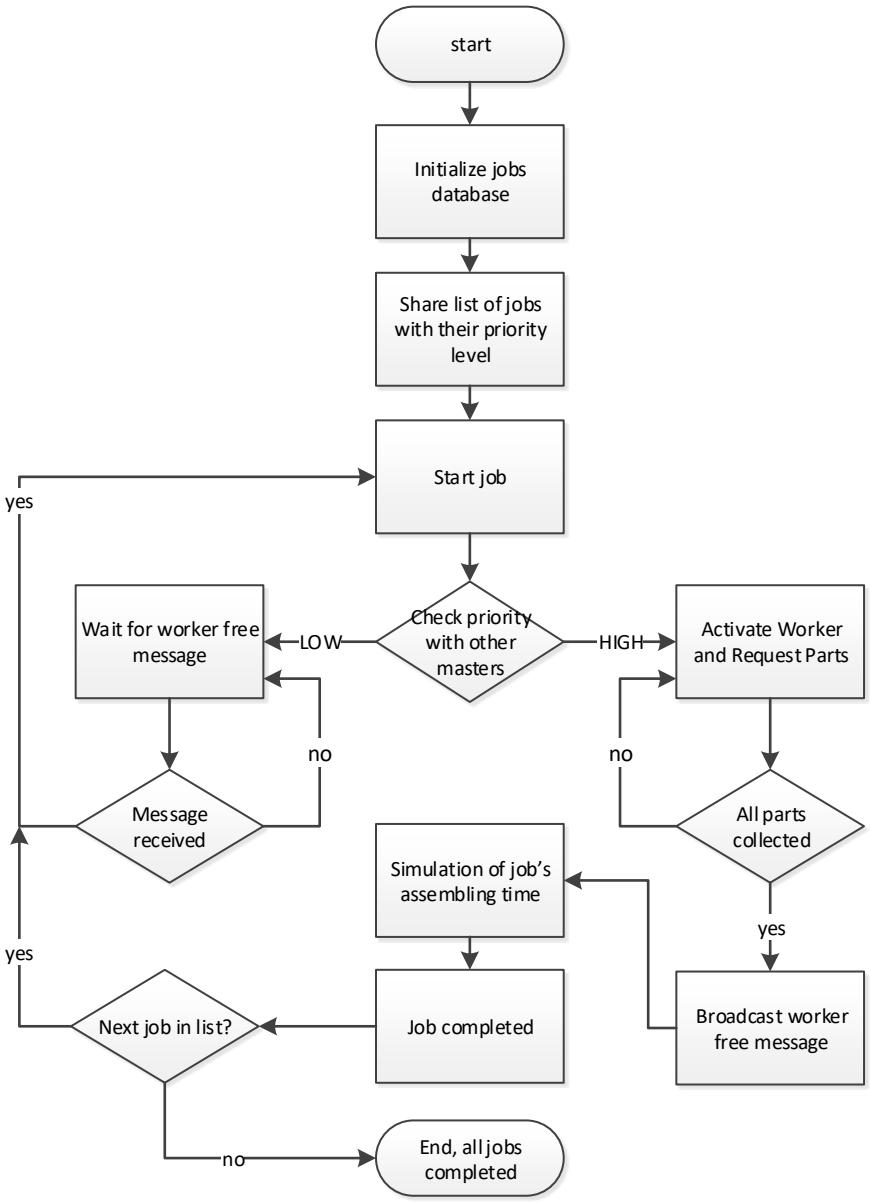


Figure 3.2 System's Flow chart

The flowchart in Figure 3.2 above illustrates the logical operation assembly line system. After the assembly line is started, it initializes the task database then share the list of all the job that it will perform with the other assembly line that are connected to the network.

A task will only start after the list sharing process is completed. Then, the system will compare the priority level of all the tasks being started on all the assembly lines in the network. If that priority is lower, it will follow the left path where it will wait for the resources to be free. If the priority is higher, it will follow the right path where it will take control over the resources until all the parts required for the job are collected. Then it will release its control over the resources, simulate the task assembly time, complete the job, and then move to the next job on the list.

3.2.1 Communication

Using Figure 1.1 System layout in chapter 1 of this document as a reference, it reveals that the communication is the main and only link between the different parts of this project (assembly lines, AGV and Storeroom). Therefore, the creation of a proper communication link is key to the success of this project. A client/server application is developed and programmed in the next units of this section. It will allow a good communication amongst the separate components.

The assembly line's application contains 2 different servers that are in constant communication with each other. The main server deal with the communication between the 2 assembly lines while the secondary sever deals with the communication between the assembly lines, the storeroom, and the AGV.

Each severs uses a pre-set range of socket to listen for new client's connection and create a communication thread that is particular to each client. This means that, if 10 clients are connected to the server, there will be 10 different communication threads running on the server. Figure 3.3 below is a visual illustration of the concept mentioned above[41].

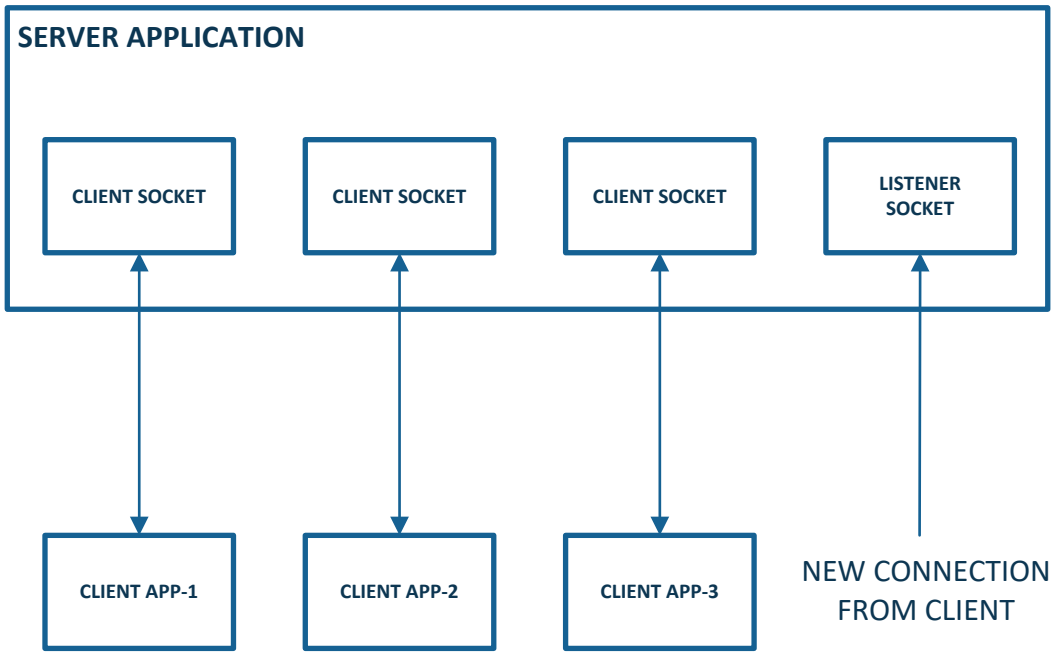


Figure 3.3 Server Application and Client sockets

The Figure 3.3 above an illustration of the operation of a client server application operation as explained in the paragraph preceding the figure.

3.2.1.1 AVG and Storeroom communication sever design

This server serves as a bridge for the communication between the different programming platforms used in this project. We used C# for the assembly line's software, LabVIEW for the AGV, Total integrated automation (TIA portal 2012) for the storeroom system.

The common factor for all these platforms, is that they can make use of a TCP/IP connection to send a byte or a sequence of bytes over a network link. Hence, this server translates messages from the assembly lines to a byte or a sequence of bytes that can be used to perform specific actions on LabVIEW (AGV) or on TIA Portal (storeroom system PLC). The bytes received from the AGV or the storeroom are also translated into a comprehensive message that can be used by the assembly lines to determine their next actions.

This server operates in the following manner:

- When started, the AVG and Storeroom communication server will initialize all of its attributes and properties
- Next it will set its IP address to the local host (computer that is running the program IP address)

- Start the listening for client connection on a specified socket range (10000). This is done so that the clients connecting to the AVG and Storeroom communication server don’t clash with the communication on the primary sever.
- After a client connection is accepted, the server assigns a socket to the communication with that client.
- Each message is converted to binary before being sent to clients.
- Each message received from the client is a single byte that is converted to a number that points to a message in the decoder array.
- The converted message is stored in a string variable that is constantly monitor by the primary sever for changes.

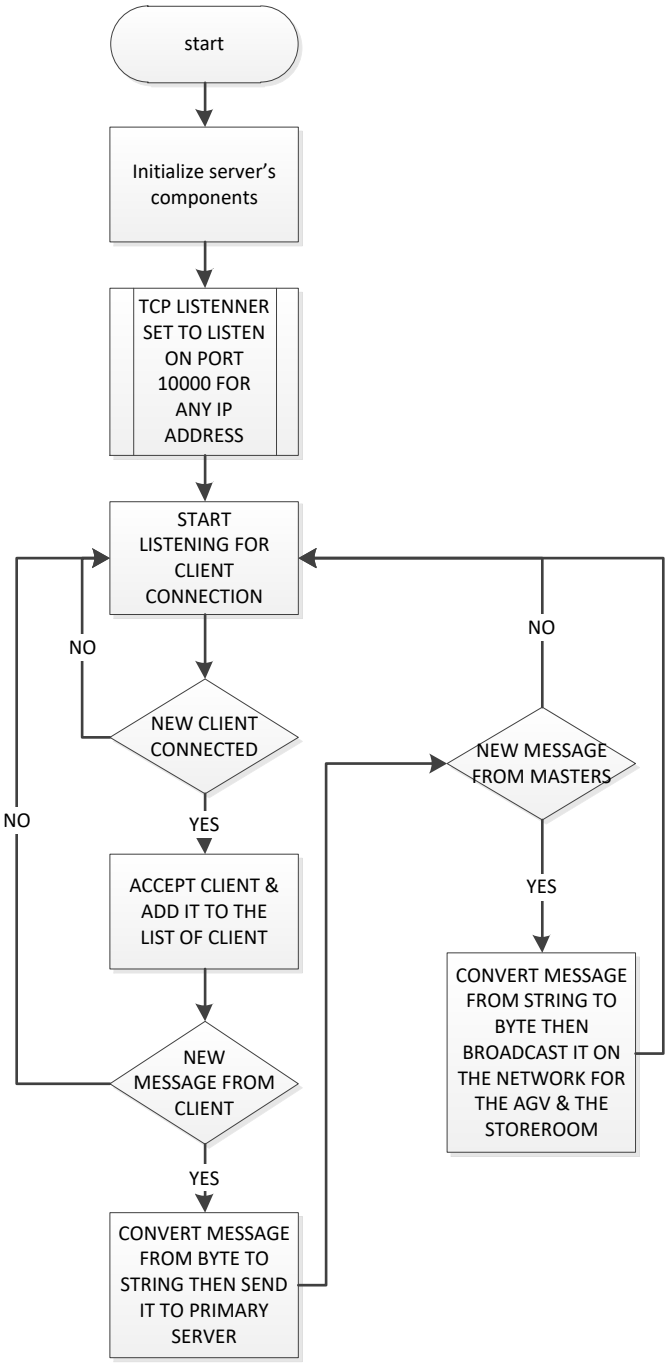


Figure 3.4 AVG and Storeroom communication server flowchart

Figure 3.4 above is the flowchart to the AVG and Storeroom communication server. After the start, it initializes the server's components, then starts the listening process on port 10000 for any IP address. If there is a new client attempting to connect to the server it will accepted and the check for messages from client. If there are no new messages, it will go back to the listening for client process until there a new client or a new message from client already connect to the server. If a new message is received from the client, it will be converted from byte to string then sent to the primary server. If a message is

received from the server, it will be converted from strings to bytes then broadcasted over the network to all clients connected.

3.2.1.2 Primary server design

The AVG and Storeroom communication sever is a slave to the primary sever. This means that, if the primary sever doesn't start, the AVG and Storeroom communication sever won't start as well because it dependent on the primary sever.

The primary server main function is to ensure proper communication between the two assembly lines. It is capable of handling a large message that is sent between the assembly lines without crashing.

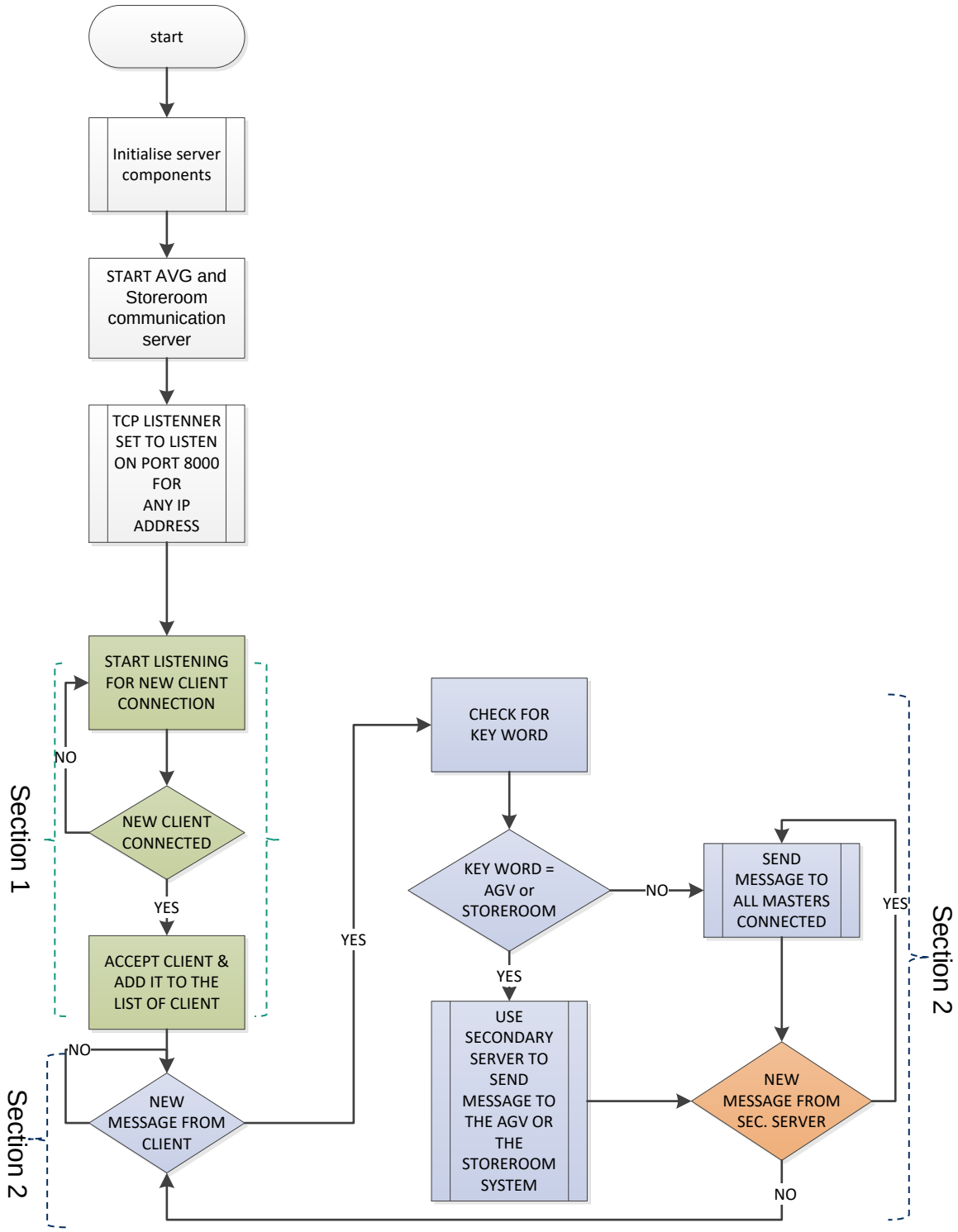


Figure 3.5 Primary server flowchart

Figure 3.5 above is the primary server's flowchart. After it starts, the server' components are initialized, the AVG and Storeroom communication server is started, then the listening process starts on port 8000 for any IP address. If there is a new client attempting to connect, it is accepted. If there is a new message from client, it is checked for key words that triggers event in the program. If the key word is AGV or storeroom, the AVG and Storeroom communication server is used to forward the message to the

AGV and the Storeroom. Otherwise, the message is broadcasted to all the client currently connected to the main server. If there is a new message from the AVG and Storeroom communication server, it will be forwarded to all the client on the main/primary server

This server monitors client connection on the socket port 8000, this is done to avoid and prevent collision with the AVG and Storeroom communication server's communication that runs on the socket port 10000.

After this server initialization, it starts the second sever and monitors the messages received by it. This server mainly consists of three main threads. Each thread controls a specific function of the server.

- The first thread, shown in section 1 in the flowchart in Figure 3.5, is the thread that controls the function of the server that is in charge of listening for client's connections.
- The second thread, shown in section 2 in the flowchart in Figure 3.5, is a thread used by the server to monitor and control the communication between the two assembly lines. In this thread, the server receives messages for assembly lines and analyses them for key words that are used to direct messages to their correct destination. This thread handles the communication between two Assembly lines, or Assembly line and AGV, or Assembly line and Storeroom.
- The third and last thread, is the thread that monitors the changes in the messages coming from the AVG and Storeroom communication server. These messages are the decrypted messages coming from the AGV or the Storeroom system.

3.2.1.3 The client

The client application connects to the server and sends messages that are broadcasted to all the other clients or devices that are connected to the server. The client is the heart of the assembly line program because that the part of the program where all the data in the system are received, analysed, and processed.

3.2.1.3.1 Client's Operation

- At its start, the client sends a request to connect to the server using its computer name (Assembly line1 or Assembly line2). The server accepts the request and broadcast the list of all clients and devices currently connected to its network.
- Next, one of two cases happen:
    1. If the new client connected is the only client on the server, the client will wait for three min for the second client or assembly line to connect or else, it will start performing all the tasks on its list.
    2. If there is already another client connected to the server, then the two clients will share the list of task that they have to perform.
- After all the tasks have been shared successfully, both assembly lines will start with the execution of their first task.

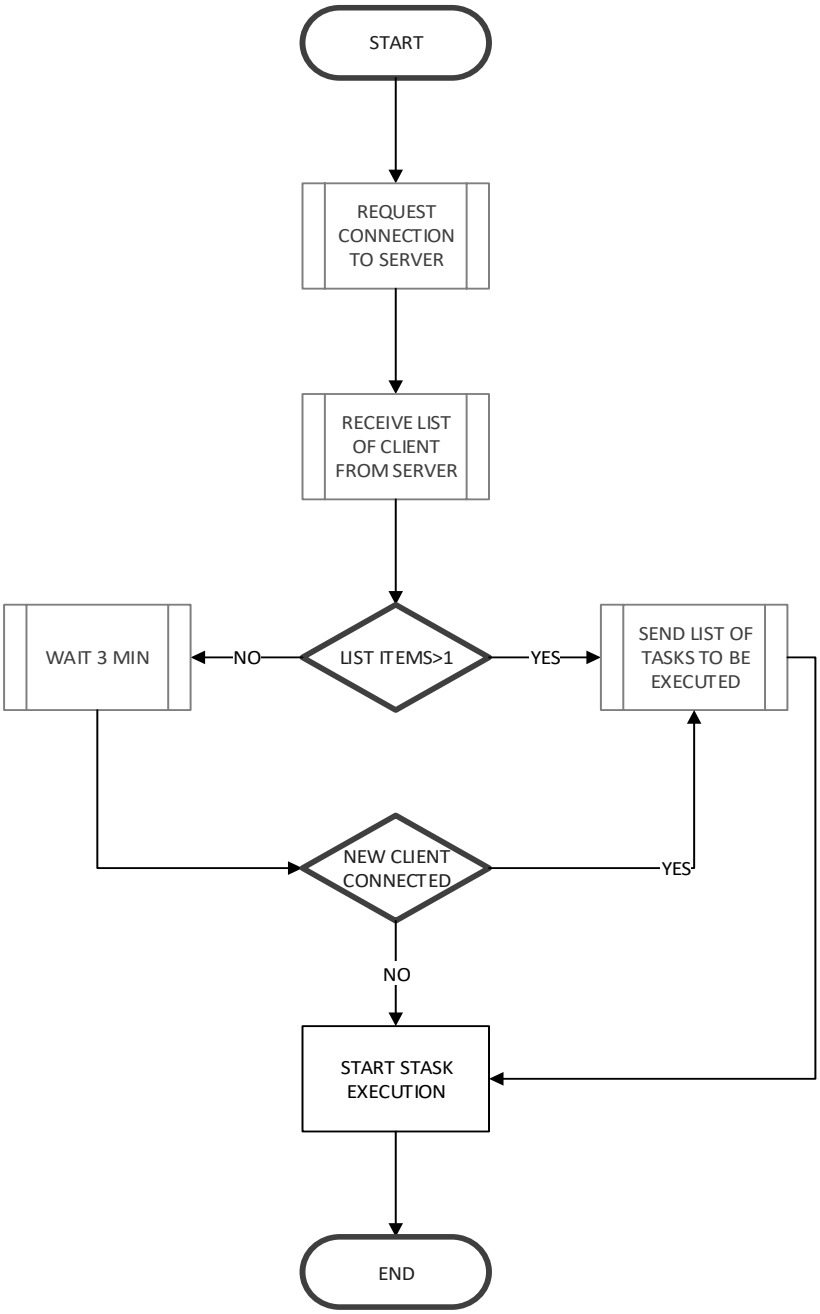


Figure 3.6 Client's basic operation flowchart

The flowchart above in Figure 3.6 illustrates the operation of the client. After its start, it requests for connection to the server, then as soon as it connected, it will receive the list of all the client connected to that sever. If the list received has more than one item, the list of tasks to be executed is sent to the other client on the server. Else, it waits for three minutes, then check if a new client to send the task list to or to start with the tasks execution.

3.2.2 Assembly line's operation

Once started, the assembly line will follow a set of steps that will allow it to perform the list of tasks or jobs that are preloaded into its memory. The assembly line's main operation happens entirely in the client part of the assembly line's program (application).

The assembly line's software is designed to be able to share its information with other assembly lines and negotiate on resource management. While analysing the system's design, three different scenarios had to be implemented in order for the assembly line to be able to work at its full potential.

The following are the three main cases:

- Scenario 1: when only one assembly line is running in the system

  In this case, the assembly line that is first to be started runs the server application that will listen and monitor new network connection. It will wait for the other assembly line to get started and connected to the server.

  However, if after three minutes the second assembly line is still not connected, this assembly line will start with the performance of the tasks in its list. It will continue going through the task list until there is a notification that the other assembly line is online and that will bring us to the second case.

- Scenario two: When both assembly lines are running at the same time

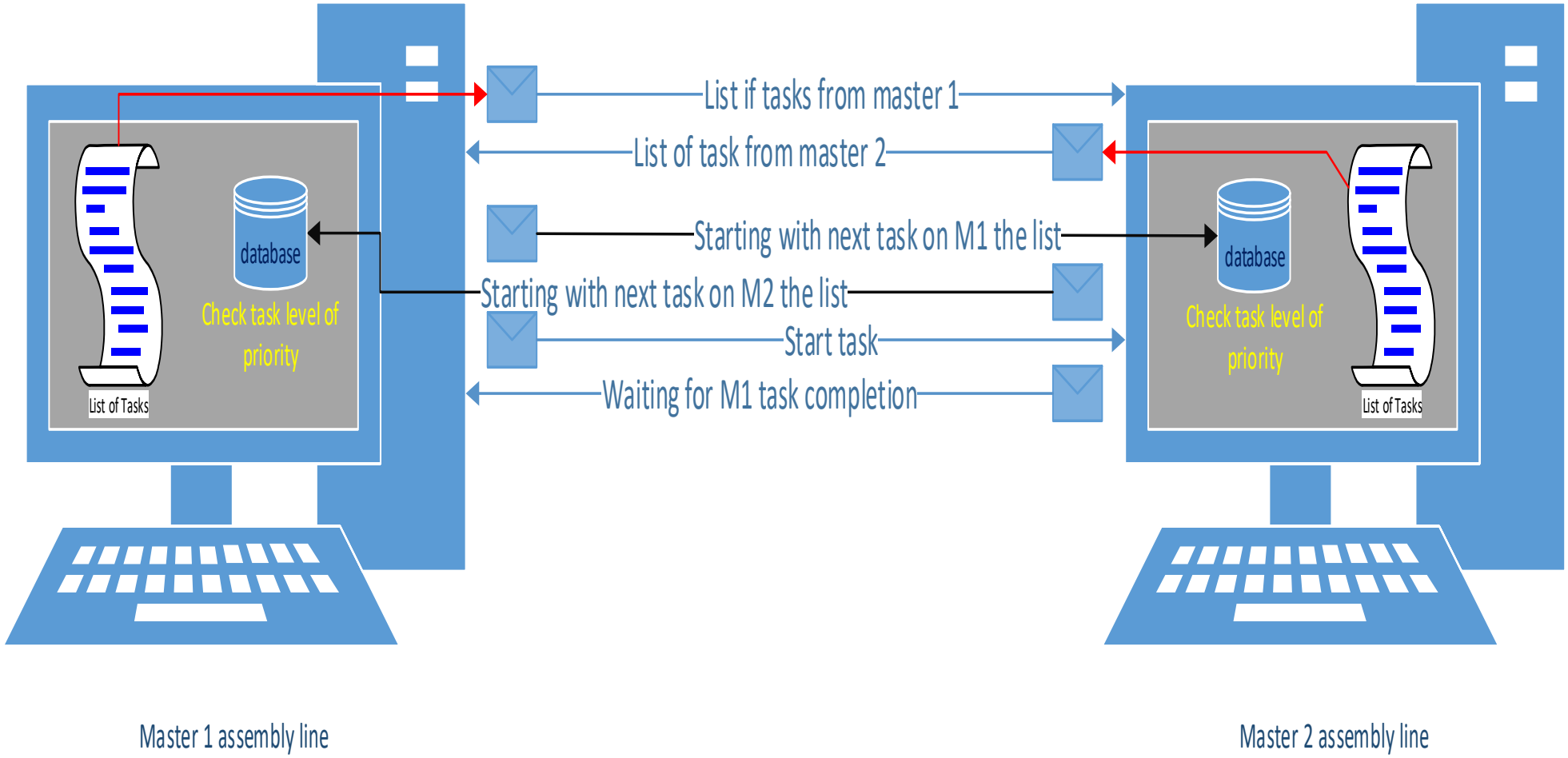


Figure 3.7 Assembly lines basic operation

Referring to Figure 3.7, in this case of the assembly line's program, it operates in the following manner:

- ✓ *Connect to a network*: the user starting the assembly line's application has the option to start a new server or to connect to an already running server using its IP Address.
- ✓ *Share a detailed list of the task or jobs that it has to perform*: this is a preloaded list of jobs that the assembly line as to perform. It will be used as a reference for the resources sharing and decision making between the assembly lines.
- ✓ *Star the execution of the tasks*: at this point, the assembly lines will each start with the execution of the tasks according to the order in which they are on the list.
- ✓ *Decide on the utilization of the resources*: each task performed by the assembly line has a level of priority based on the time that it requires to be assembled. The job taking the longest time will have the highest priority level.
- ✓ *Use/wait for resources*: depending on the level of priority of the task that the assembly line is starting, it can take control of the resources (AGV and Storeroom system) or wait for the other assembly line to be done using the resources.

- Scenario three: when one of the assembly lines completes all the tasks on its list and the other one is left with more than one task on its list:

  In this case, the assembly line that is not yet done with its tasks will share its remaining task with the assembly line that is done.

The client makes use of a state machine to execute different functions of the assembly line. Each case in Figure 3.8 below will run indefinitely until the condition to trigger the start of the next case is met. These cases allow the assembly line to function in a sequential flow. Because some of the functions of the assembly line are crucial, this state machine forces the execution of specified cases to avoid crashing or code malfunction.

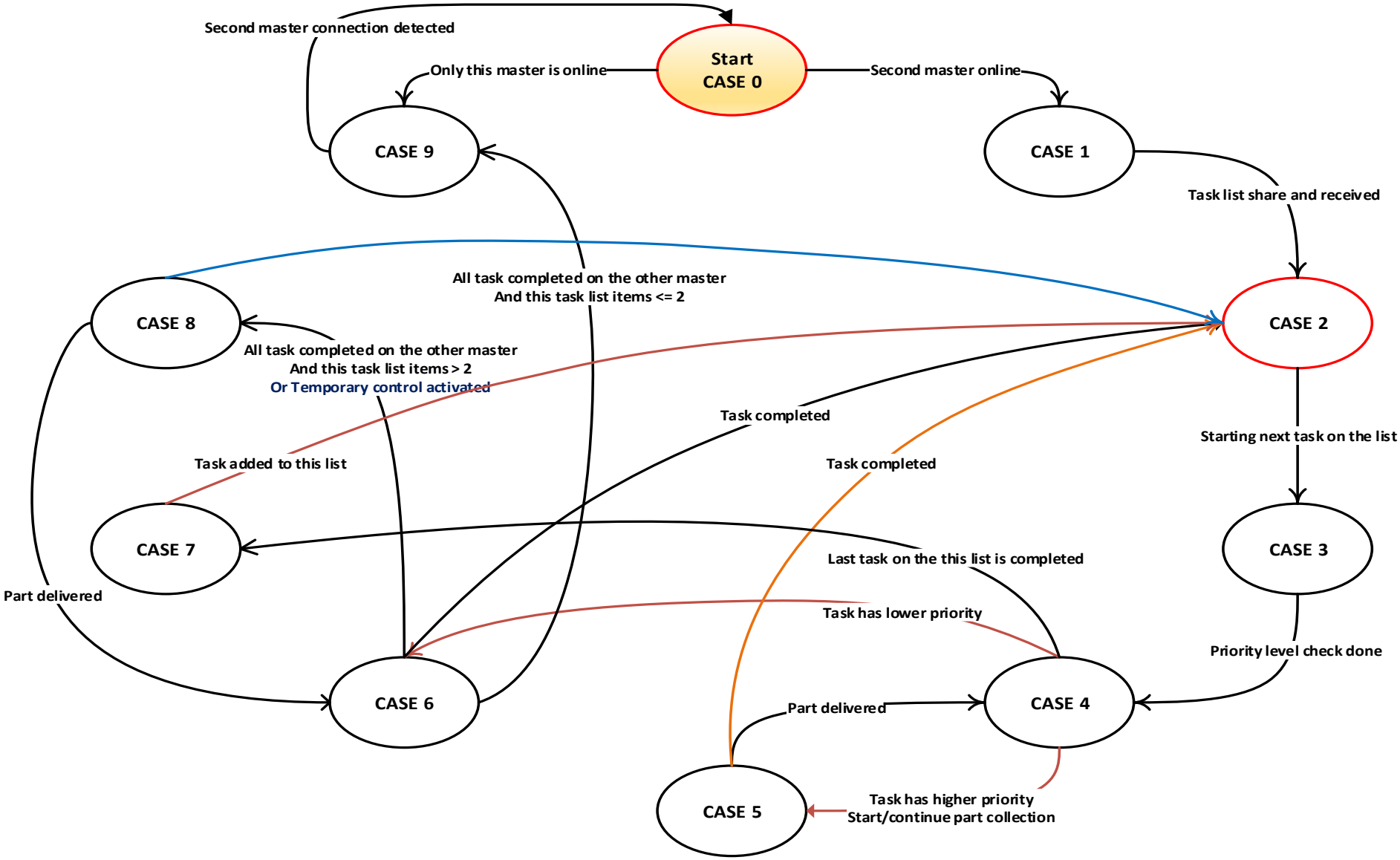


Figure 3.8 Client State Machine

Figure 3.8 above displays the client's state machine. It shows the ten cases used and their relationships that are further discussed in the next section below.

3.2.2.1 State machine cases overview

3.2.2.1.1 Case 0 and 9

Case 0 is the starting case of the state machine shown in Figure 3.8. Its purpose is to wait and monitor the server's messages to know if the other assembly line is connected to the network. Depending on which condition is met, this case triggers 2 other cases. Case 9 is a case in which only one assembly line is running on the network or is left with uncompleted task/jobs on its list. Figure 3.9 below is a flowchart describing the steps followed in case 0 and 9.

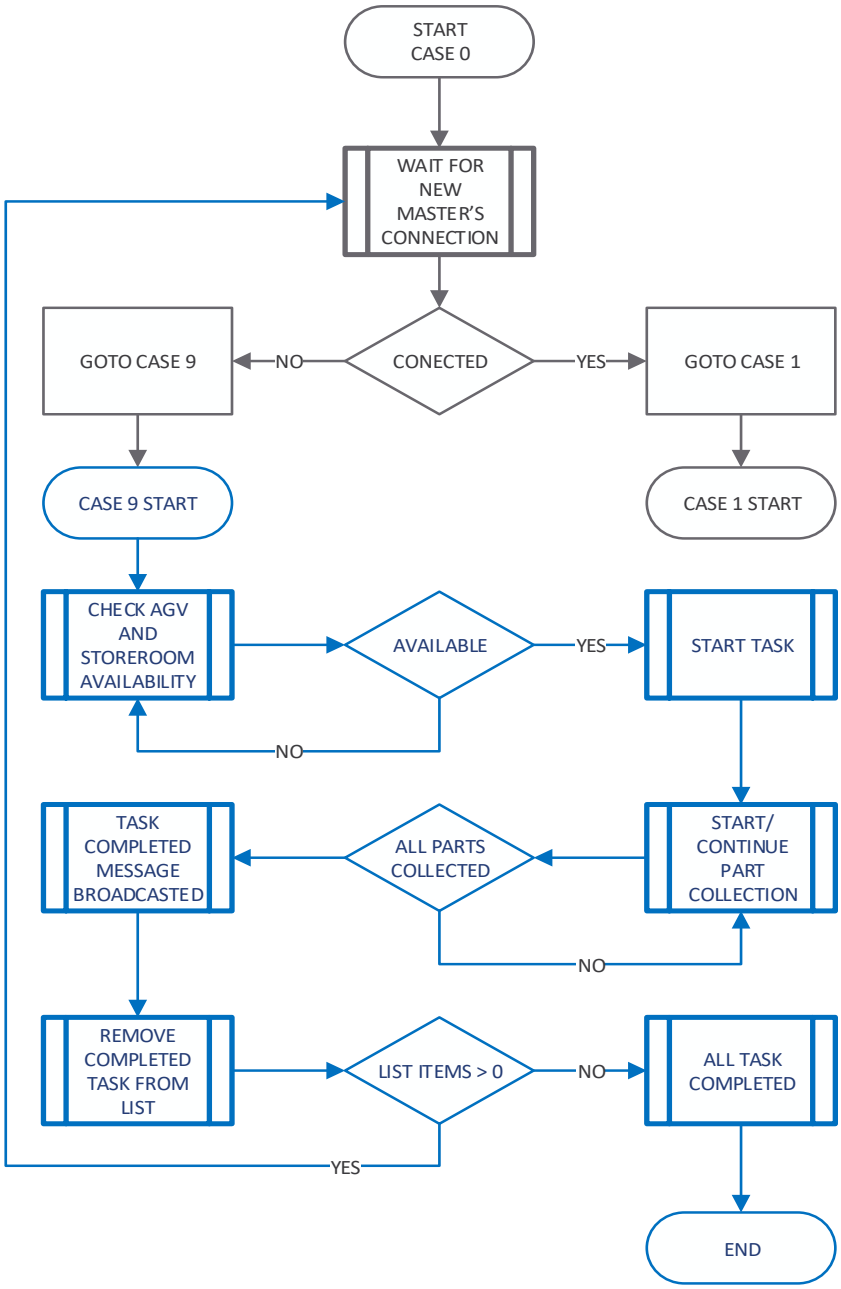


Figure 3.9 Case 0 & 9 Flowchart

Figure 3.9 is the flowchart of the first and ninth cases of the state machine. It shows that after the start of the case 0, the assembly line waits for the server to let it know if there no other assembly lines connected to the same server. This results to the start of case 1 or 9 depending on whether there are other assembly lines online. Case 9 start by checking the availability of the AGV and the storeroom, and then starts with the execution of the task in the list. The task's parts collection stars right after that and does not stop until all the needed parts are collected from the storeroom by the AGV. The process done by case 9 is executed from the start of the case until all the task in the list

are completed. However, this execution is interrupted at the instant that this assembly line is no longer the only assembly line connected to the server

3.2.2.1.2 Case 1

This case controls the function of the assembly line in which the list of tasks is shared between the assembly lines. The list received will be stored and used as a reference to know what task will be performed by the other assembly line.

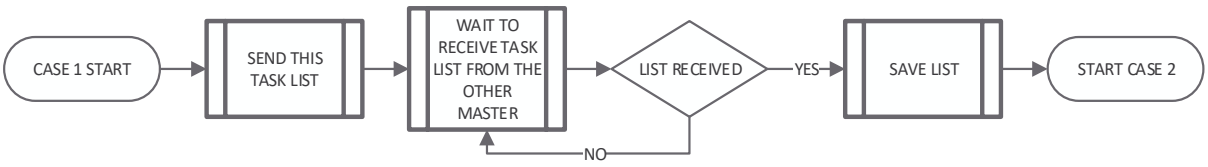


Figure 3.10 Case 1 flowchart

Figure 3.10 above is the flowchart of the case 1, it sends the list of tasks from the current assembly line to the other mast, then waits the receive the list from the other assembly line, save it and start the next case in the state machine.

3.2.2.1.3 Case 2

This case is used to broadcast a message to indicate that this assembly line is about to start a new task from its list.

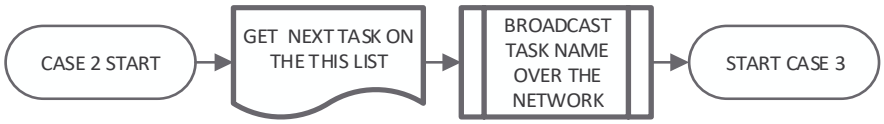


Figure 3.11 Case 2 flowchart

Figure 3.11 above shows the flowchart of case 2. When started, it selects the next task on the list, shares it over the network, and then starts the next case.

3.2.2.1.4 Case 3

This assembly line makes use of this particular case to check for the level of priority of the task that it is about to start compared to the one that is starting on the other assembly line. Figure 3.12 shows a step by step flowchart of the operation of this case. It starts by receiving the name of the task being started on the other assembly line, then uses the task description database to get all details about that task. The task's details contain the priority level of the task. A priority level is a number ranging from 1 to 4 that is used by the two assembly lines to know which one of them is executing the task with the highest priority.

While programming this case, provisions were made to solve the following conditions for the priority level check:

- ✓ What should happen when Assembly line 1 is starting the task with the highest priority (refer to Figure 3.12 section 1): if this assembly line has the highest priority task, the state machine will continue to case 4 directly.
- ✓ What should happen when Assembly line 2 is starting the task with the highest priority (refer to Figure 3.12 section 2): if this assembly line has the lowest priority task, the state machine will also continue to case 4 directly.
- ✓ And what should happen when they are starting the same task (same priority level) (refer to Figure 3.12 section 3): if both assembly lines have the same level of priority, case 3 is rerouted to the section 3 on the flowchart below. They will both randomly generate and share a number between 1 and 10, and the assembly line with the highest generated number will take priority over the resources usage while the other will wait.

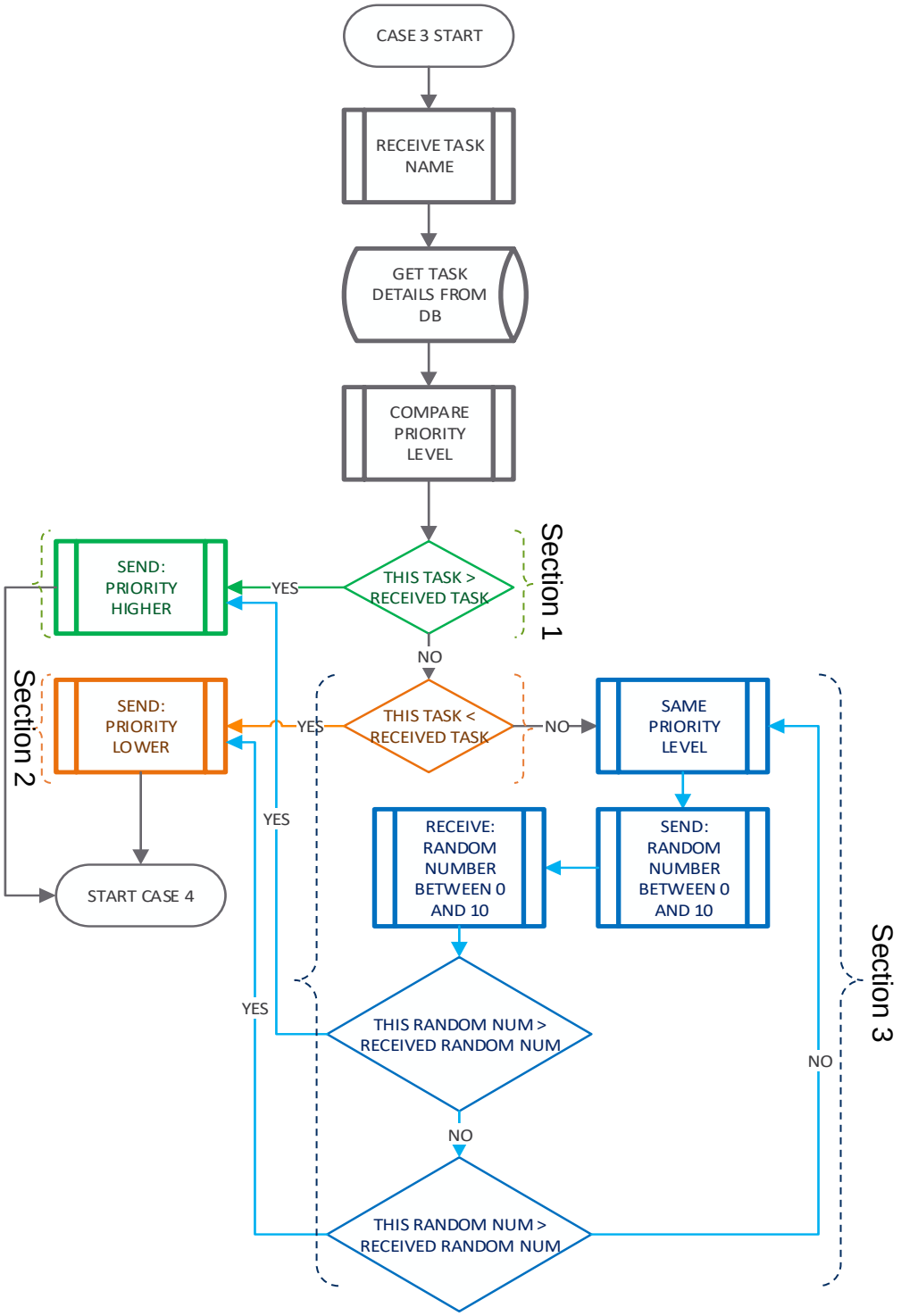


Figure 3.12 Case 3 flowchart

Figure 3.12 above is the flowchart of case 3 and it operates as explained I the paragraph above.

3.2.2.1.5 Case 4

Referring to Figure 3.8, you can see that this case is started from case 3 or 5 depending on the current operation that is being performed by the program. This section will go through what happens when this case is started from case 3, then from case 5 referring to Figure 3.13.

- When started from case 3:

  At the end of the execution of case 3, the result of the priority check process is shared as a message between the 2 assembly lines. The path followed by this case is determined by this result. Therefore, if the result message says higher, it means that the other assembly line will have control over the resources until the completion of its task. This assembly line will go to case 6 and wait for the task completion notification from the other assembly line. If the result message says lower, this assembly line will take control over the resources and start with the execution of the task.

  Before the collection of part starts, a few conditions have to be met first.

  1. The AGV must be online
  2. The AGV current location must be the storeroom
  3. The number of parts collected must be less than the total number of parts needed for this task.

  Only when all these conditions are met will the assembly line send a message to the AGV to collect whatever part it needs. After the part collection message is sent, the case 4 stops and the program will move to case 5. Refer to Figure 3.13 section 1

- When started from case 5:

  When the program goes from case 5 to case 4, they will be no need for priority check because case 5 can only be executed when this task has the highest priority and parts are being collected. Refer to Figure 3.8, Figure 3.13 section 1 and Figure 3.14.

The part collection and task execution process will go back and forth between case 4 and 5 until the task is completed (refer to Figure 3.13 section 1 and Figure 3.14) or all the tasks on this list are completed (refer to Figure 3.13 section 2)

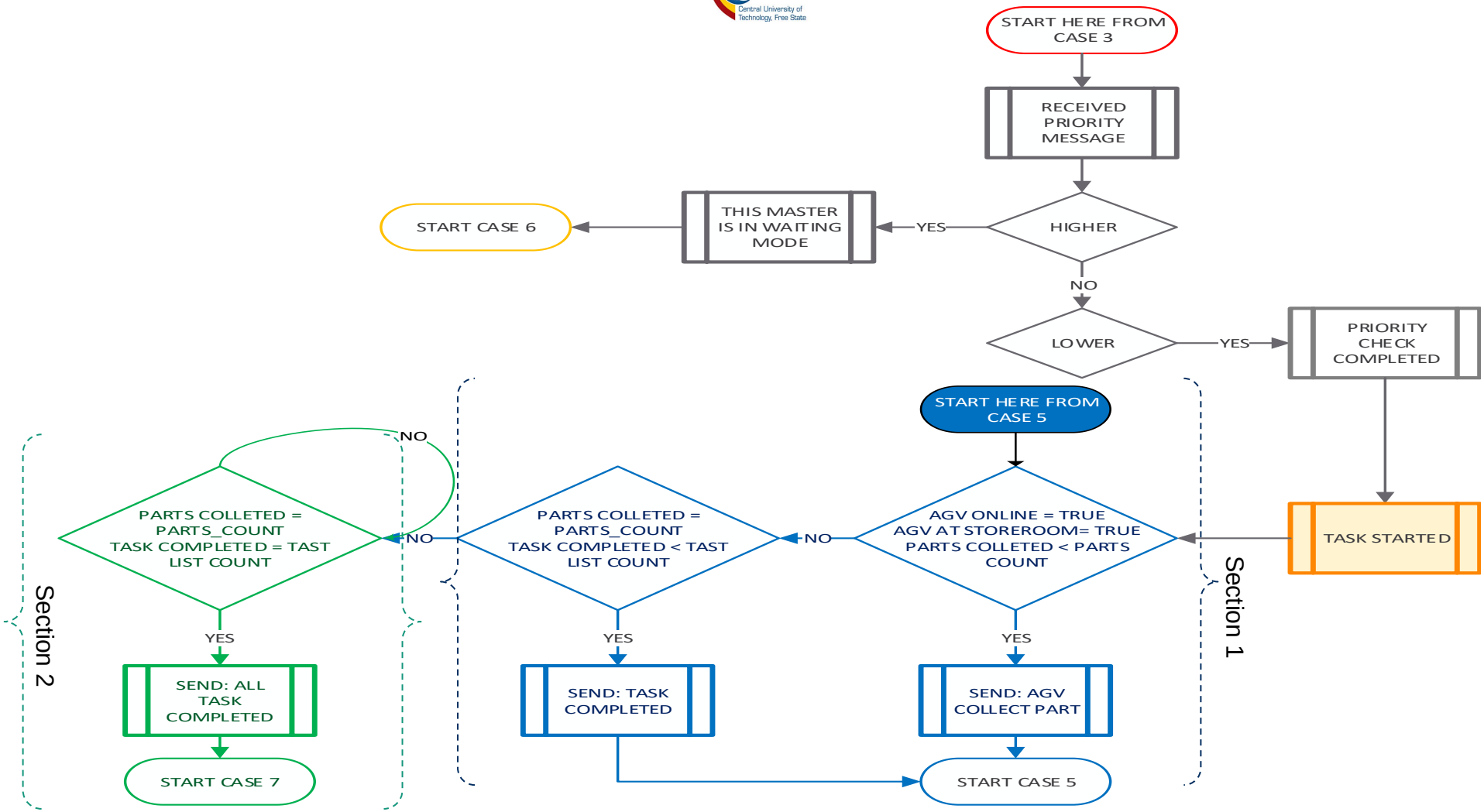


Figure 3.13 Case 4 flowchart

Figure 3.13 above is the flowchart for case 4 and its functions are explained in the paragraph above.

3.2.2.1.6 Case 5

The purpose of this case is to monitor the progress of the part collection. Each task has a list of part that it requires to be completed. This simply points the program to the part that has to be collected next or to the next task that has to be executed. Refer to Figure 3.14 for a descriptive flowchart of the base operation of this case.

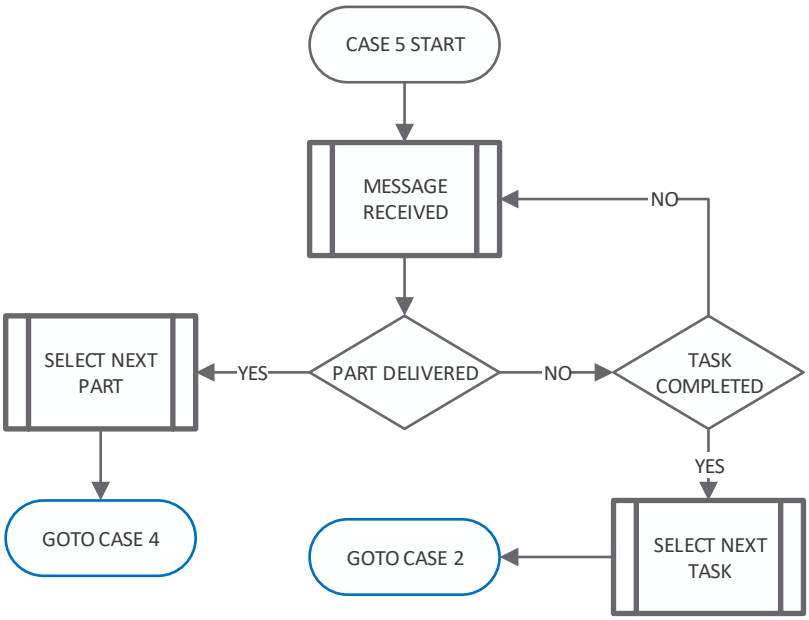


Figure 3.14 Case 5 flowchart

Figure 3.14 displays the operation flowchart of case 5. It monitors the messages received from the server to know if a part is delivered to the current assembly line. It the message says that a part is delivered, the next part is select, and the state machine goes back to case 4. If the message says task completed, the next task on the list is selected and the state machine will go to case 2.

3.2.2.1.7 Case 6

This case gets started from case 4 when the current assembly line is starting a task that has a lower priority level compared to the one started on the other assembly line. It is used to wait and monitor the progress of the task that is being process on the other assembly line. Depending on the message received, this case will follow one of two paths:

1. Task completed: this message is sent by the other assembly line to notify that the task in progress has been completed. Refer to Figure 3.15 section 1. The case will stop the wait and go to case 2.
2. All task completed: this indicates that the other assembly line has completed all the task on its list and it is entirely free. Refer to Figure 3.15 section 2. The wait will stop

and will go to case 8 if the number of remaining task on this list is greater than 2 or proceed to case 9 if otherwise.

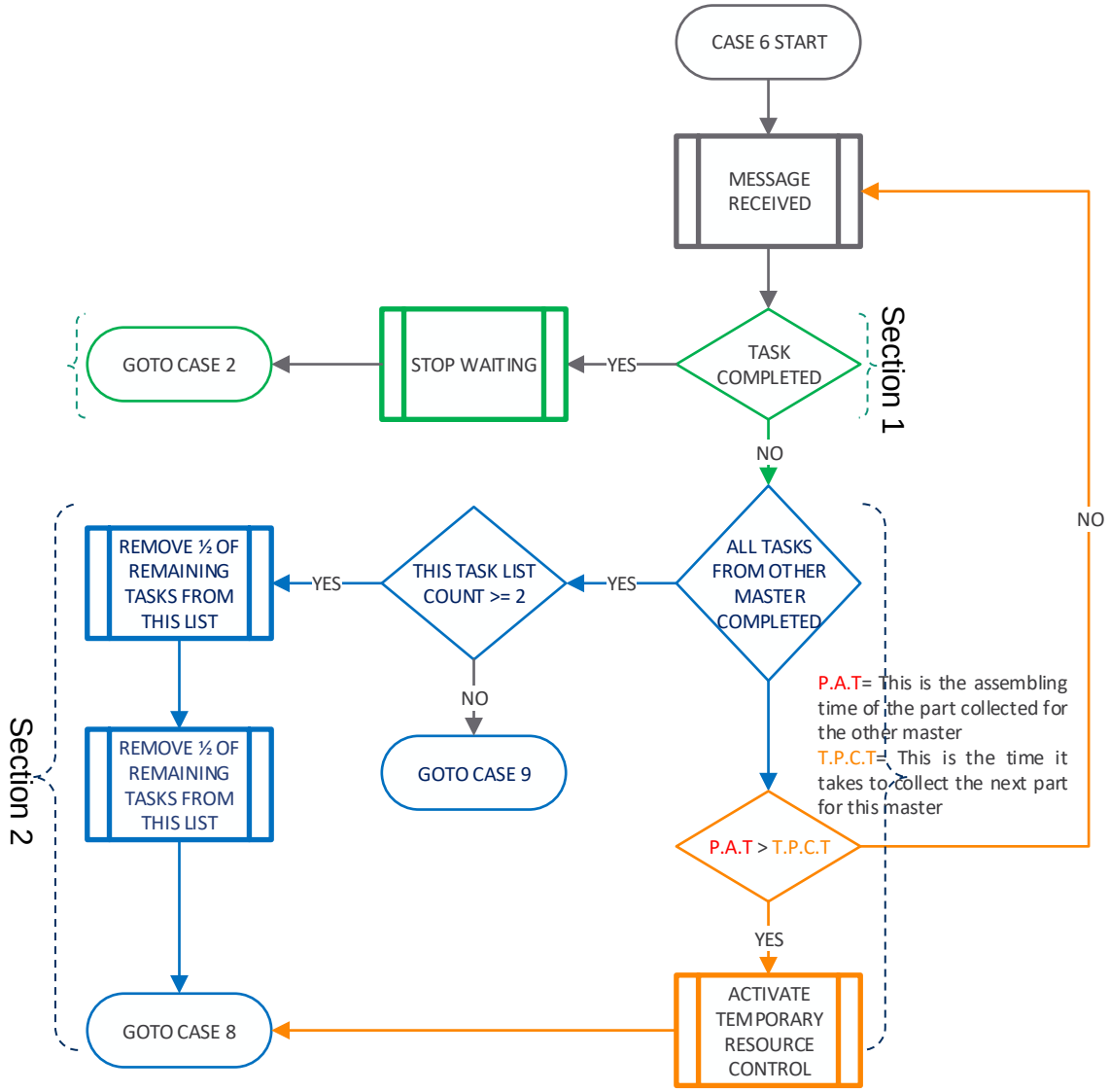


Figure 3.15 Case 6 flowchart

Figure 3.15 above is the flowchart of case 6 and it operates as explained in the paragraph preceding it.

3.2.2.1.8 Case 7

From case 4, when all the tasks on this list have been completed, the program jumps to this case to check if the other assembly line has more than 2 task left on its list. The program in this case monitors messages from the other assembly line that will say how many tasks are left. This assembly line will then add half of those remaining tasks to its own list. It will go to case 2 to start to completion of the added tasks. Refer to Figure 3.16 below.

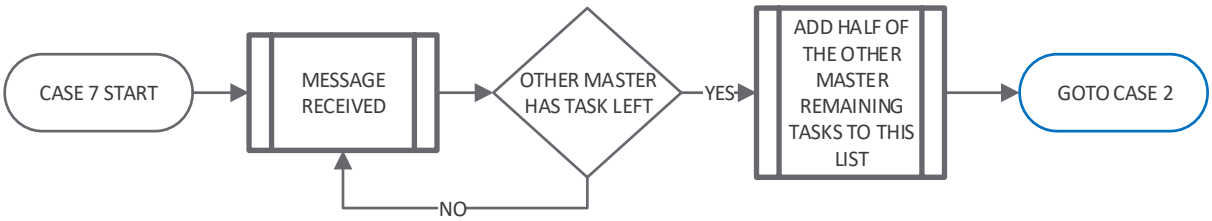


Figure 3.16 Case 7 flowchart

Figure 3.16 above is the flowchart of case 7 and it operates as explained in the paragraph above it.

3.2.2.1.9 Case 8

From case 6, if all the tasks on the other assembly line's list are completed and this assembly line has more than 2 task left on its list, referring to Figure 3.15 section 2, this assembly line will send the quantity (number) of task left to the other assembly line and then come to this case to wait for a message from the other assembly line that half of this assembly line's remaining task have been successfully added to the other assembly line's list. The program will then jump to case 2 to start with task execution. Refer to Figure 3.17 below.

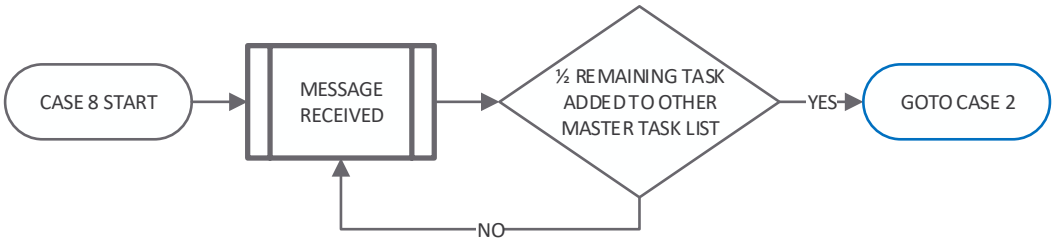


Figure 3.17 Case 8 flowchart

Figure 3.17 above is the flowchart of case 8 and it operates as explained in the paragraph above it.

3.3 The worker

The system's layout shown in Figure 1.1 of the introductory chapter shows an AGV that collects parts from the storeroom and delivers them to the assembly lines. The NI LabVIEW Robotics Starter Kit was selected to be used as the AGV. It will be fitted with line follower sensors and programmed to follow a black line on a white surface.

"It is designed to help prototyping an autonomous system and quickly get familiar with the capabilities of LabVIEW Robotics software and NI reconfigurable I/O (RIO) hardware" [42].

NI LabVIEW Robotics Starter Kit is a fully assembled mobile robot base starter kit with an ultrasonic sensor, encoders, DC motors, and a 12V battery. It has a controller based on NI Single-Board RIO with real-time decision making, FPGA-based I/O processing and analogue and digital I/O on a single board. It can easily connect to a variety of robotic sensors and actuators [43].

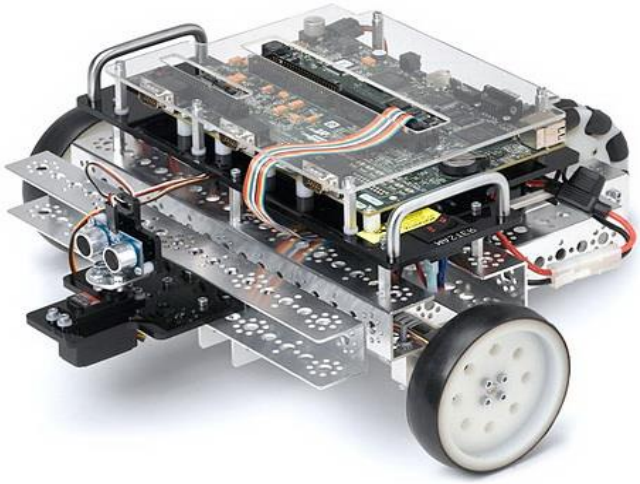


Figure 3.18 NI LabVIEW Robotics Starter Kit[44]

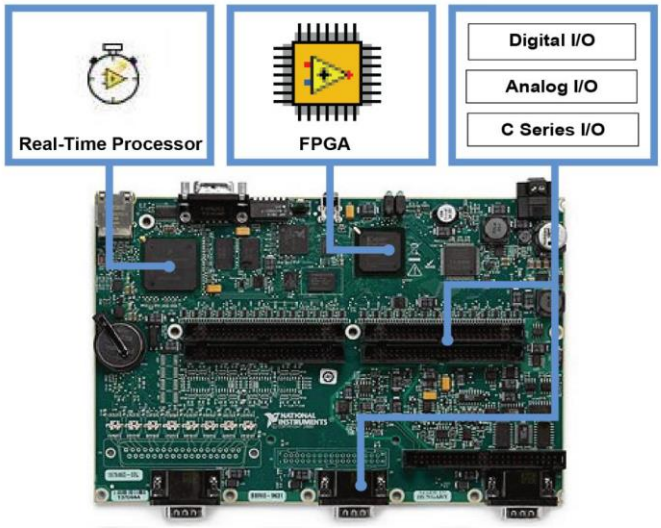


Figure 3.19 the 9632 NI Single-Board RIO includes a real-time processor, FPGA, and built-in digital and analogue I/O [45]

Figure 3.18 and Figure 3.19 show respectively the Robotic starter kit and its controller board. Figure 3.19 also shows the different parts that makes the RIO board (the processor, FPGA, and all the inputs and outputs)

The AGV will be autonomously capable of sensing its environment, process the information, and make decisions depending on the commands received from the assembly lines (assembly lines). The process followed by the worker is divided into four different steps:

- Communication
- Sensing
- Decision
- Action

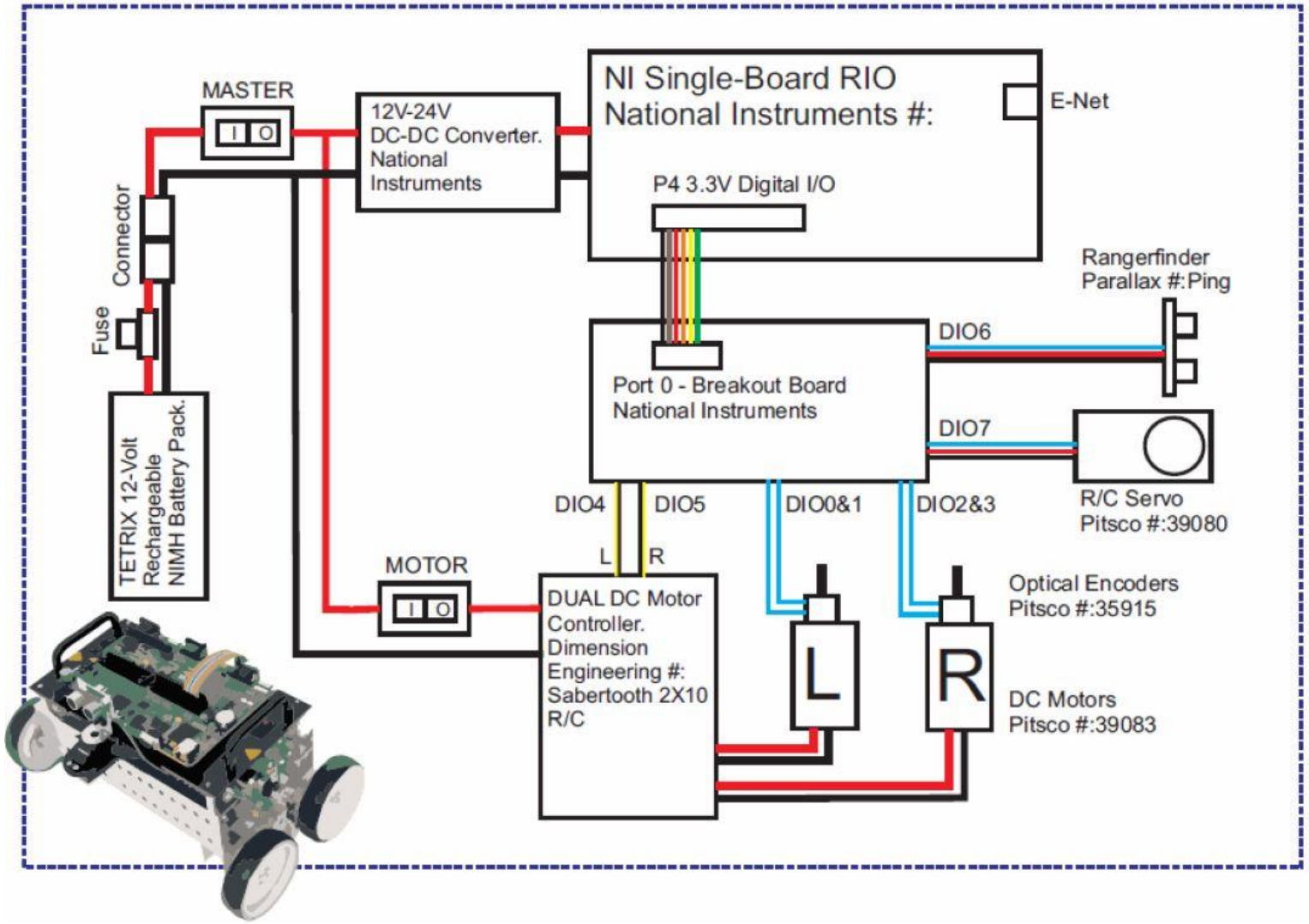


Figure 3.20 LabVIEW robotics Starter kit (connections diagram)

Figure 3.20 shows the connection diagram of the robotic starter kit, starting from the battery to the proximity sensor, the servo and the motors.

3.3.1 Communication

The worker (AGV) is in constant communication with the assembly line, sending information about its location and receiving instructions about what it has to do and where to go. The communication is done through a local area network in which the entire the system is connected.

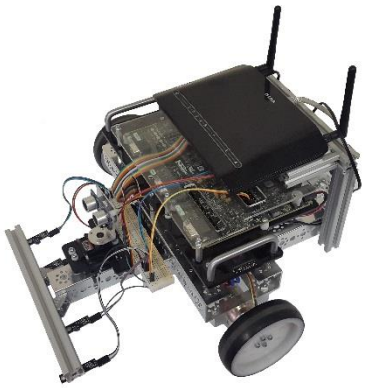


Figure 3.21 LabVIEW robotic starter kit fitted with Wi-Fi router

Figure 3.21 above show the project AGV design. It is a LabVIEW robotic starter kit fitted with Wi-Fi router at the top and four digital line follower sensors at the bottom front.

The NI LabVIEW Robotics Starter Kit has an Ethernet port that allows it to perform TCP/IP communication over a network of NI (National Instrument) device. Therefore, the communication can also be established between the AGV (NI LabVIEW Robotics Starter Kit) and a network of computer. A server/client communication application is implemented between the assembly line and worker, where the assembly line is the server and the worker is the client.

TCP/IP communication requires a server and a client. In this case, the worker (AGV) is the client and assembly line is the server. Communication is initiated by the AGV by requesting access to the server at a specified IP address (10.0.0.3) and Port number (10000).

The server and client communicate by sending one byte of data over the network link, which in turn is decrypted into commends or message that must be executed.

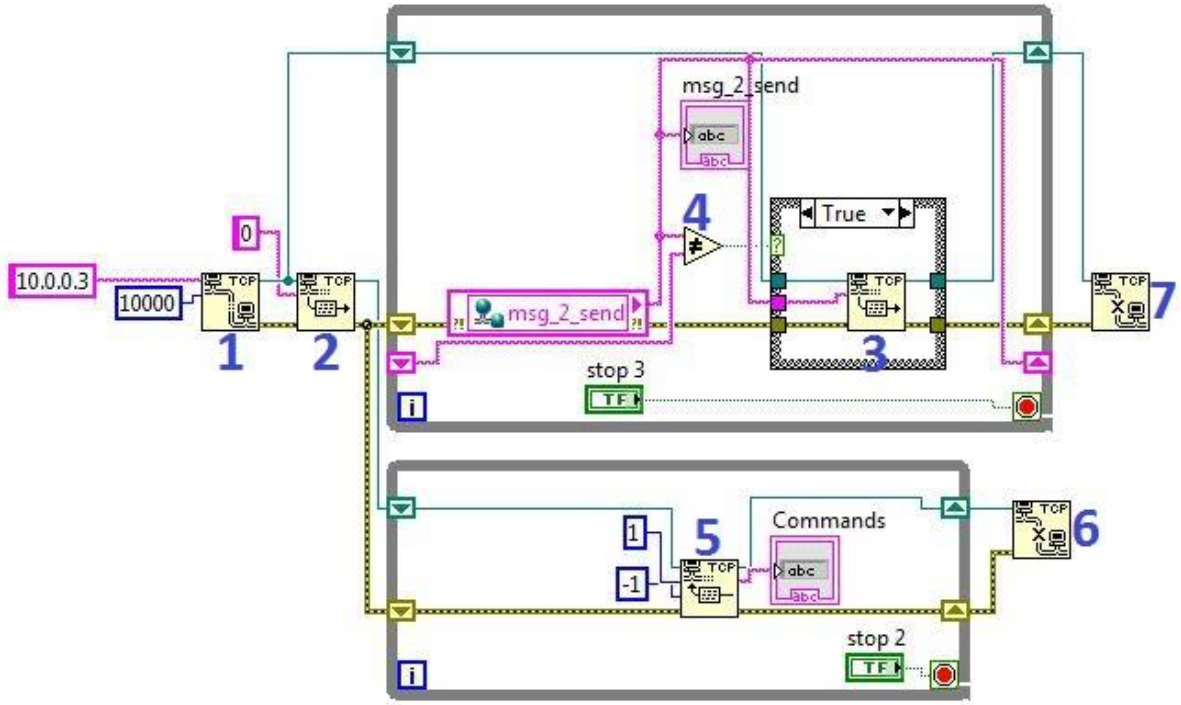


Figure 3.22 AGV communication function diagram

Figure 3.22 above is the communication function diagram for the AGV. The description and purposes of the labels 1 to 7 are described below:

- 1: Is a TCP OPEN function
- 2: is a TCP WRITE function, it is used to write messages onto the network stream. Table 3.1 below better explain the communication process and messages between the Assembly line and the AGV.
- 3: is also a TCP WRITE function, but it is utilized in a loop to send constant messages to the server every time that a change is made to the client status. For example: if the AGV is in the storeroom or traveling over the line…
- 4: is a comparator that is used to compare the message that is about to be sent to the previous message and returns a TRUE value if the two messages are different from each other. That value goes to a case statement where the message is sent when it receives TRUE and nothing is sent when the value is FALSE. This is done to avoid sending the same message more than once over the network because of the endless loop in which the TCP Write function is running.
- 5: is a TCP read function that constantly reads commands over the network for the client (AGV) to execute
- 6 & 7: are TCP close connection that is used to terminate the connection to the server.

Figure 3.23 below shows the flowchart of the AGV communication function diagram shown in Figure 3.22

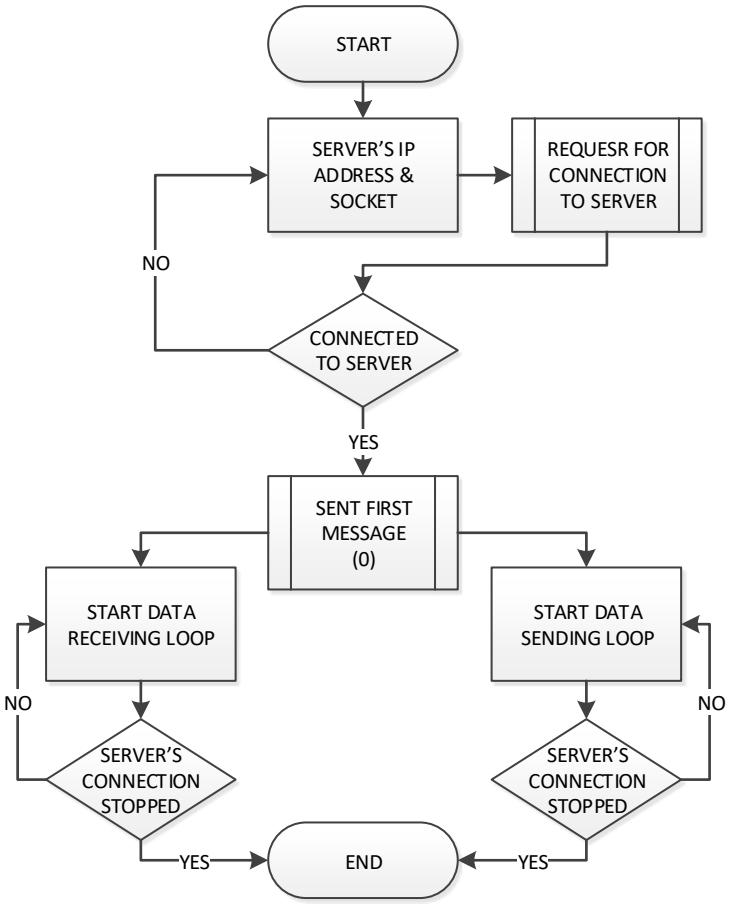


Figure 3.23 Communication flowchart

Figure 3.23 above is the communication flowchart related the function diagram shown in Figure 3.22. The first step after the start is to set the server's IP address and the socket range, then request for a connection to the server. When the AGV connects to the server, it sends a message to server that lets all the connected devices know that the AGV is online. Afterward, two loops are simultaneously started. One on the left for messages reception, and the other to the right for message broadcasting.

The TCP Read function require that the number of bytes to read from the network to be set to a constant number. For example, if the byte to read input is set to ten and the message received only has five bytes, nothing will be read until it receives ten bytes and the application will stop running until all ten bytes are read. On the other hand, if the byte to read is set to five and the message received contains ten bytes, only the first five bytes will be read and the rest will be discarded.

So to avoid that problem, the TCP Read function is set only to read one byte that is then decrypted to a command using Table 3.1 below. Table 3.2 is utilized by the server to decrypt messages from the client (AGV worker). This allows for a straightforward communication between the server and the client without any crashing or miscomprehension of messages.

Table 3.1 Server's messages Decryption

Commands from Server	Decryptions	comments
0	Stop	This command from the server stops all actions being done by the AGV at the moment that it is received.
1	Go to Assembly line 1 location	This command instructs the AGV to go from the storeroom to the site of assembly line 1 to deliver the requested part.
2	Go to Assembly line 2 location	This command instructs the AGV to go from the storeroom the location of assembly line 2 to deliver the required part.
3	Go back to storeroom from assembly line 1	The command is used to tell the AGV to return to the storeroom after delivering a part to assembly line1
4	Go back to storeroom from assembly line 2	The command is used to tell the AGV to go back to the storeroom after delivering a part to assembly line2

Communication is a two ways process. Therefore, the server needs to know if its messages are received by the client and executed before sending the next command. Hence, all the messages coming from the client needs to be decrypted and processed to know when the client is ready to execute the next command. Table 3.2 below is the decryption table used for messages coming from the client to the server.

Table 3.2 Client's messages Decryption

Messages from Client	Decryption	comments
0	AGV online	This message is sent to let the server know that client is online and ready to receive commands.
1	Part Collected	This tells the server that the part requested by the assembly line has been collected from successfully from the storeroom and that the AGV is ready to deliver that part to the required assembly line.
2	Part delivered to Assembly line1	This tells the server that the part needed has been delivered to assembly line1 and that it is ready to return to the storeroom.
3	Part delivered to Assembly line2	This tells the server that the part needed has been delivered to assembly line2 and that it is ready to go back to the storeroom.
4	AGV@StoreRoom	This message informs the server that the AGV is back to the storeroom location after delivering a part to either assembly line1 or assembly line2.

3.3.2 Sensing

The AGV uses infrared sensors to follow a black line on a white surface. They are sensors are digital and give an output of 1 when they are pointing on the white surface and 0 when they are pointing on the black line.

These sensors have a very low range in which they can operate successfully. That is why they will be placed at the bottom of the AGV at 10mm from the surface that they have to read.

The AGV is required to follow a black line on a bright surface, hence, we made use of Figure 3.24 below that shows the circuit diagram of the digital IR sensors used in this project. The sensor requires the components listed in Table 3.3 to build them. The operation of the sensor shows that it gives out an output of 1 when it is pointing to the bright surface and 0 when it is pointing to the black line.

Table 3.3 List of components for IR sensor

Components	Quantity	Comments
IR LED	4	Emitter of the IR radiations
Photodiode	4	Receiver of the IR radiations
150Ω resistor	8	Protection resistors
10KΩ resistor	4	
10KΩ variable resistor	4	This resistor is used to set the sensitivity of the sensor
LM358M op-amp	4	This device has been used to convert the analog signal from the sensor to the digital 1 or 0 used by the AGV

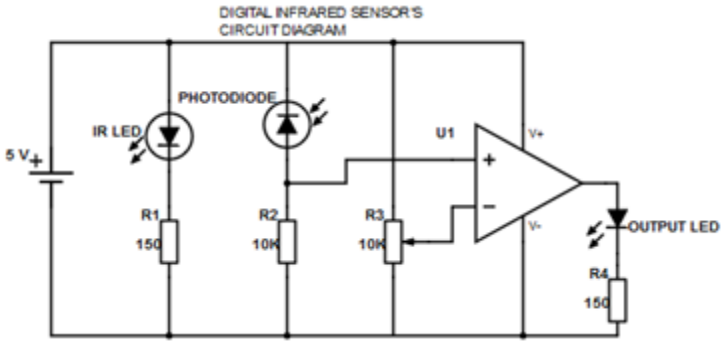


Figure 3.24 Digital IR sensor circuit

Figure 3.24 above, is the circuit diagram for the digital line follower sensors. it is composed of an infrared LED that serves as the emitter, a photodiode that is used as the receiver, a network of resistors (2 x 150 and 2 x 10 000 ohms), and an operational amplifier (Op-Amp) used as a comparator.

In this project we made use of four sensors and each sensor plays a specific role in the decision that the AGV make while following the line.

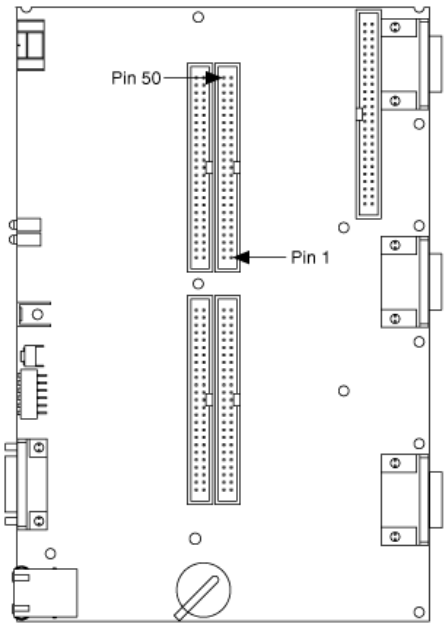
Table 3.4 Sensors roles

Label	Description	Role	comments
R1	Right sensor 1	Line following sensor	This sensor is about 4cm on the **Right** of the black. It has a default value of **1** because it should always point to the bright reflecting surface. Every time that its value changes to **0**, the AGV turns **Right**.
L1	Left sensor 1	Line following sensor	This sensor is about 4cm on the **Left** of the black. It has a default value of **1** because it should always point to the bright reflecting surface. Every time that its value changes to **0**, the AGV turns **Left**.
R2	Right sensor 2	Right check points counter sensor	This sensor counts the check points on the **Right** hand side of the line. Refer to D. Decision on page 69 of this section for more explanations
L2	Left sensor 2	Left check points counter sensor	This sensor counts the check points on the **Left** hand side of the line. Refer to D. Decision on page 69 of this section for more explanations

Each sensor forms an independent circuit that needs to be the connected to the AGV. As shown in Figure 3.25 and Figure 3.26, the AGV has parallel connectors with reserved pins in which a digital circuit can be connected.

3.3.2.1 ***Connecting sensor to AGV***

This project made use of parallel connector 5 on the NI rio board that controls the AGV to connect our four sensors. Refer to Figure 3.25…



D GND	50	49	Port4/DIO8
5 V	48	47	Port4/DIO7
D GND	46	45	Port4/DIO6
5 V	44	43	Port4/DIO5
D GND	42	41	Port4/DIO4
D GND	40	39	Port4/DIO3
D GND	38	37	Port4/DIO2
D GND	36	35	Port4/DIO1
D GND	34	33	Port4/DIO0
Port4/DIOCTL	32	31	Port4/DIO9
D GND	30	29	Port3/DIO8
D GND	28	27	Port3/DIO7
D GND	26	25	Port3/DIO6
D GND	24	23	Port3/DIO5
D GND	22	21	Port3/DIO4
D GND	20	19	Port3/DIO3
D GND	18	17	Port3/DIO2
D GND	16	15	Port3/DIO1
D GND	14	13	Port3/DIO0
Port3/DIOCTL	12	11	Port3/DIO9
D GND	10	9	Port7/DIO3
D GND	8	7	Port7/DIO2
D GND	6	5	Port7/DIO1
Port7/DIO9	4	3	Port7/DIO0
Port7/DIOCTL	2	1	D GND

Figure 3.25 Connector P5, 3.3 V Digital I/O on NI sbRIO-9632/9632XT [46]

Each sensor has three points that need to be connected to the AGV:

- VCC: the voltage supply of the sensor. If not connected the sensor will not work
- GND: the ground on the sensor circuit.
- Output: this is a digital bit that will allow the AGV to monitor the status of the sensor.

Table 3.5 Sensors Connections

Label	Pins on sensor	Connection to P5	Pin number on P5
R1	VCC	5V	50
	GND	D GND	48
	OUTPUT	Port3/DIO2	17
R2	VCC	5V	50
	GND	D GND	48
	OUTPUT	Port3/DIO3	19
L1	VCC	5V	50
	GND	D GND	48
	OUTPUT	Port3/DIO1	15
L2	VCC	5V	50
	GND	D GND	48
	OUTPUT	Port3/DIO0	13

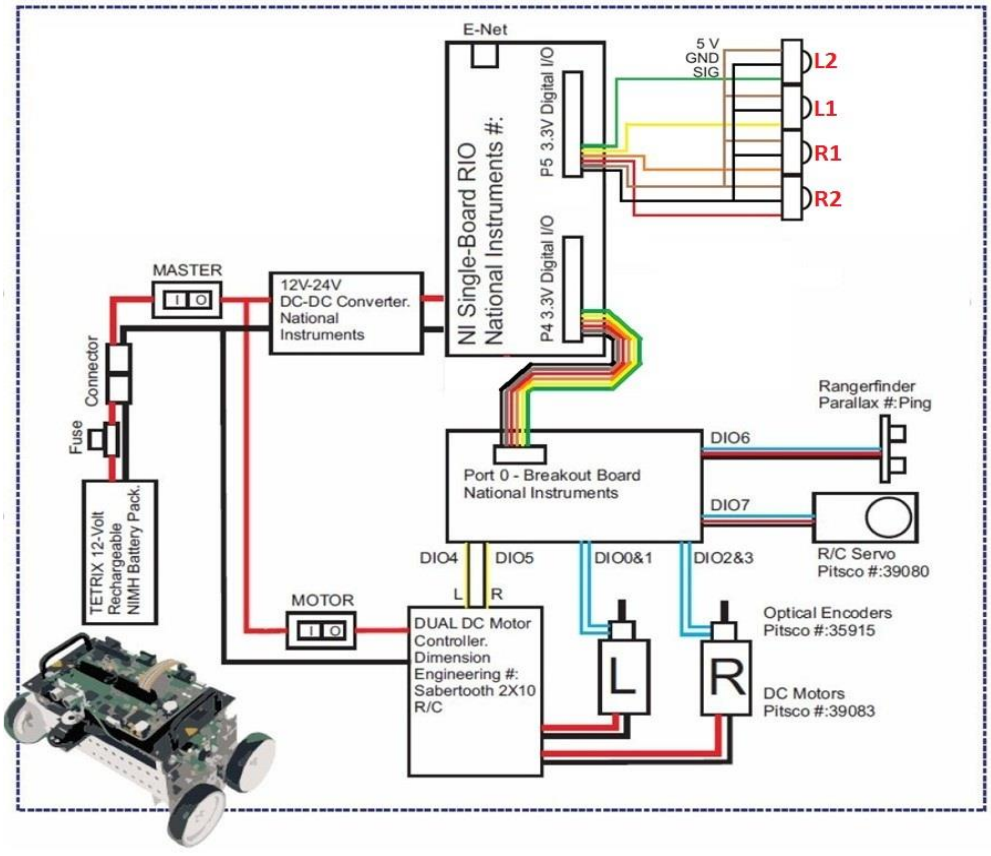


Figure 3.26 LabVIEW robotics Starter kit (block diagram) with sensors connection

Figure 3.26 shows the block diagram of the NI LabVIEW robotic starter kit (AGV or Worker) with the sensors connected to it.

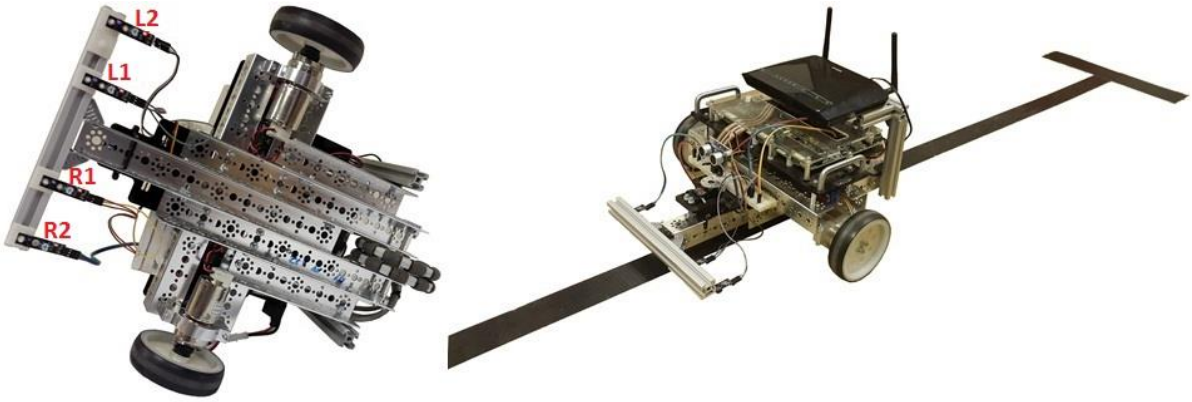


Figure 3.27 AGV with line follower sensors

Figure 3.27 is a picture of the physical AGV showing the sensors connected to the AGV.

3.3.3 Decision and Action

This section of the worker describes how the commands from the assembly lines and the inputs from the sensors are converted into decisions and actions that the AGV has to perform and follow.

The first step of this section is to highlight all the possible actions that the AGV can perform. The second step will be the decisions that the AGV can take based on the commands form the assembly lines and the input of the sensor.

3.3.3.1 Step 1: Action

Five different type of actions can be performed by the AGV and all these actions depend on the status of the motors.

Table 3.6 List of AGV actions

Actions	**Right motor**	**Left motor**	**comments**
Stop	No motion	No motions	
Move forward	Turn clockwise	Turn clockwise	
Move backward	Turn anticlockwise	Turn anticlockwise	
Turn Right	Turn clockwise	Turn anticlockwise or no motion	
Turn Left	Turn anticlockwise or no motion	Turn clockwise	

The AGV is programmed using NI LabVIEW. LabVIEW developers have provided an entire section reserved for robotic. That part of the software allows users to develop applications for a robotic starter kit. They provided users with some examples for some of the core function that the robotic starter kit can perform

In this project we made use of the example where it shows how to control the motors and monitor the ultrasonic sensor. To find these examples, you can follow the steps shown below starting from Figure 3.28 to Figure 3.36.

LabVIEW robotics must be installed on the windows pc in for it to work.

Referring to Figure 3.28, the first step is to create a new robotic project using LabVIEW robotics 2014. Note this process may differ depending on the version of LabVIEW that the user may have.

- Open LabVIEW robotics 2015,
- Under create project, double click on robotics project.

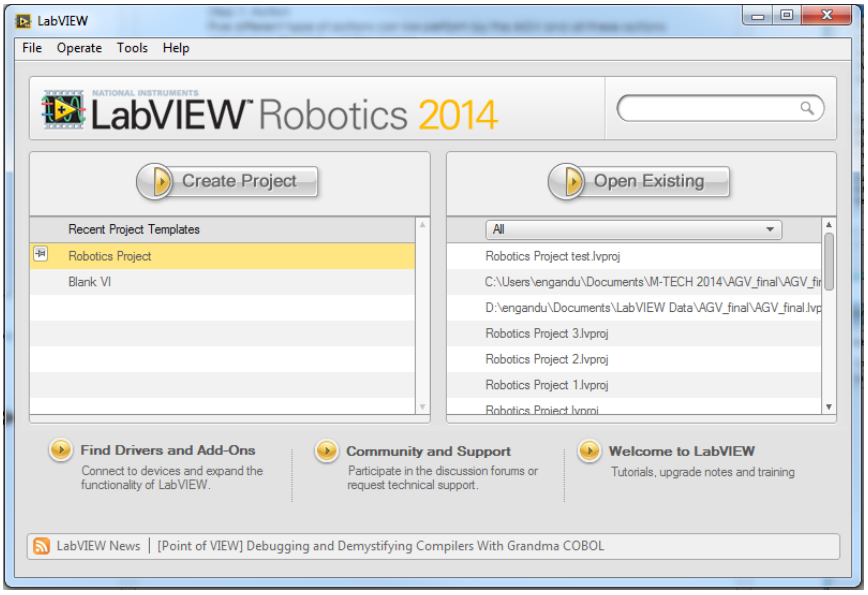


Figure 3.28 labView robotics 2014

- The new robotics project shown in Figure 3.29 should appear that will allow you to select the type of project that you would like to create.

- Different types of robotics project may be set up at this point depending on the platform that is being used. In project we made use of the **Robotics Starter Kit 2.0**. Therefore, in the project type list I have selected **Robotics Starter Kit 2.0** as shown in Figure 3.29 below

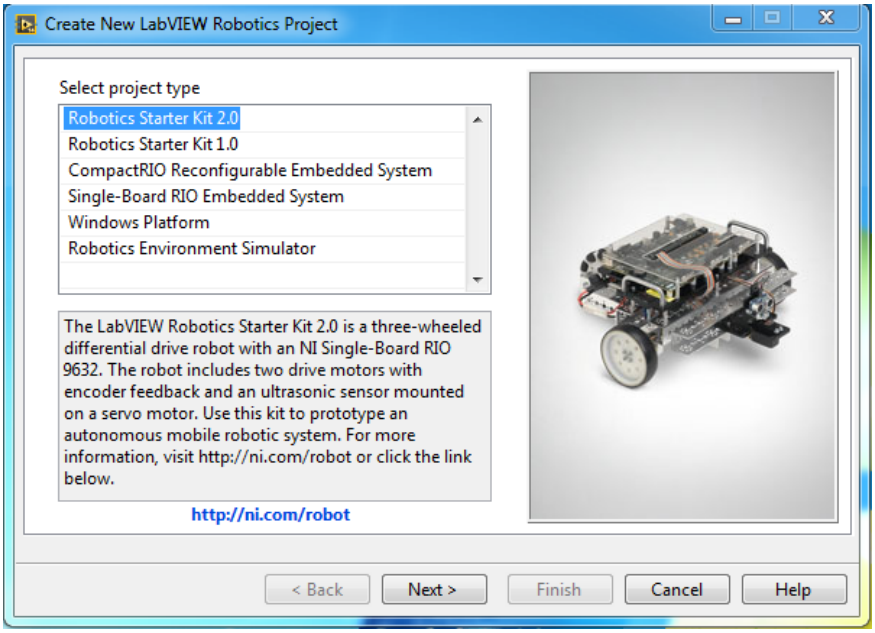


Figure 3.29 New labView robotics project window

- For this next step, **Robotics Starter Kit 2.0** should be connected to the computer using an Ethernet cab. Its IP address needs to be entered in the **controller IP address** text box. Refer to Figure 3.30 below. Then click on next.

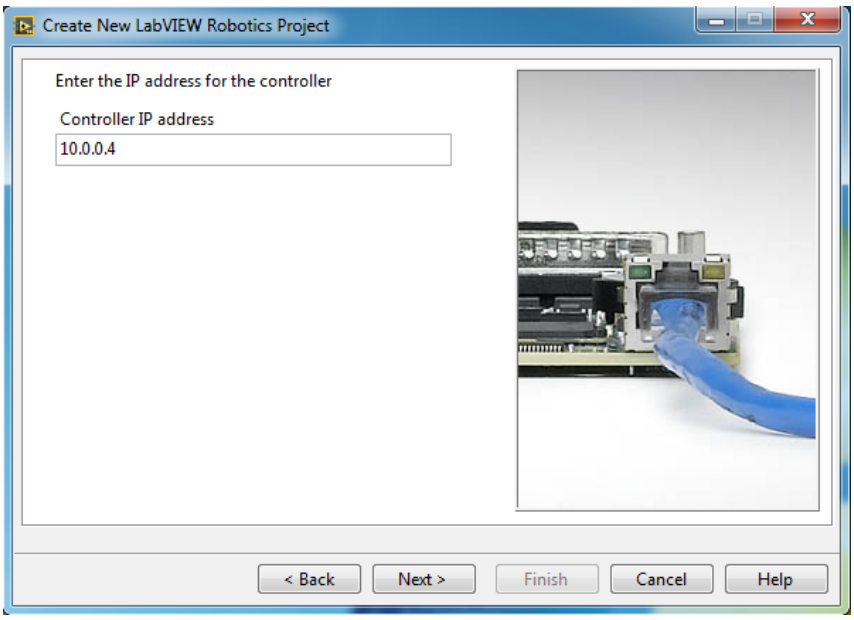


Figure 3.30 Controller IP address window

- In Figure 3.31 you will notice that there are different type of architecture that can use as for starting up with the programming of a LabVIEW robotics starter kit. These architectures are skeleton program that can be modified to suit the users need. For this project, the best suited skeleton is the "**starter kit 2.0 single loop**" as selected in Figure 3.31 below. This specific architecture gives you access and control over the motors speed and at the same time monitors the ultrasonic sensor data. Click next to move to the next step of the project creation.

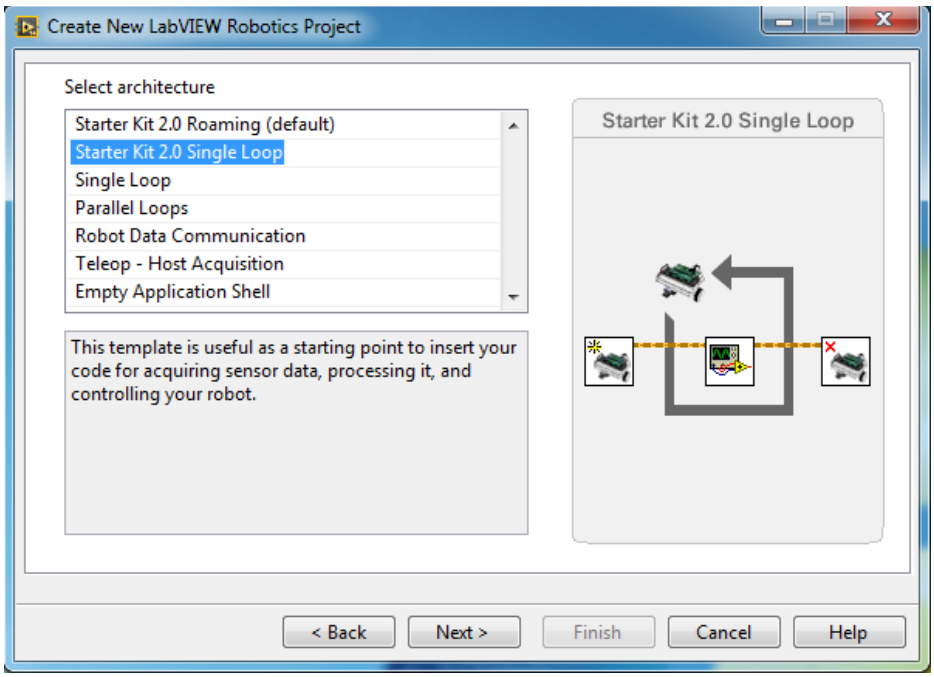


Figure 3.31 robotics architecture window

- At this point every necessary settings have been made and the only left is to enter the name of your projects and select the destination in which your project will be saved then click finish. Referring to Figure 3.32 below.

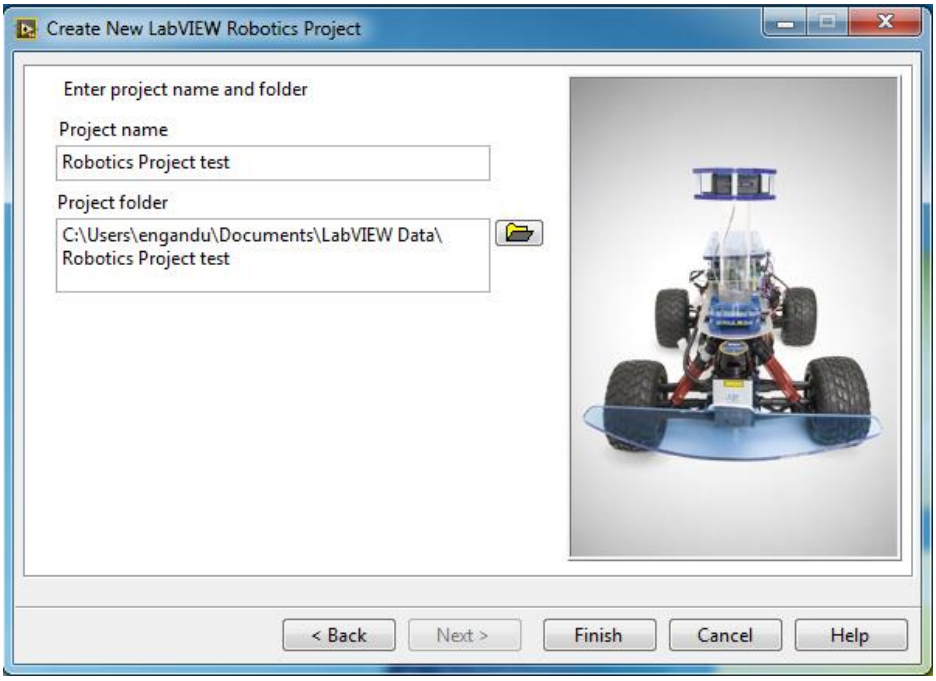


Figure 3.32project name and destination window

- The project explorer window in Figure 3.33, allows you to navigate and view different parts of your project. Double click on the **main.vi** in **starter kit 2.0 sbRio (10.0.0.4)** to see the front panel of the project shown in Figure 3.34.

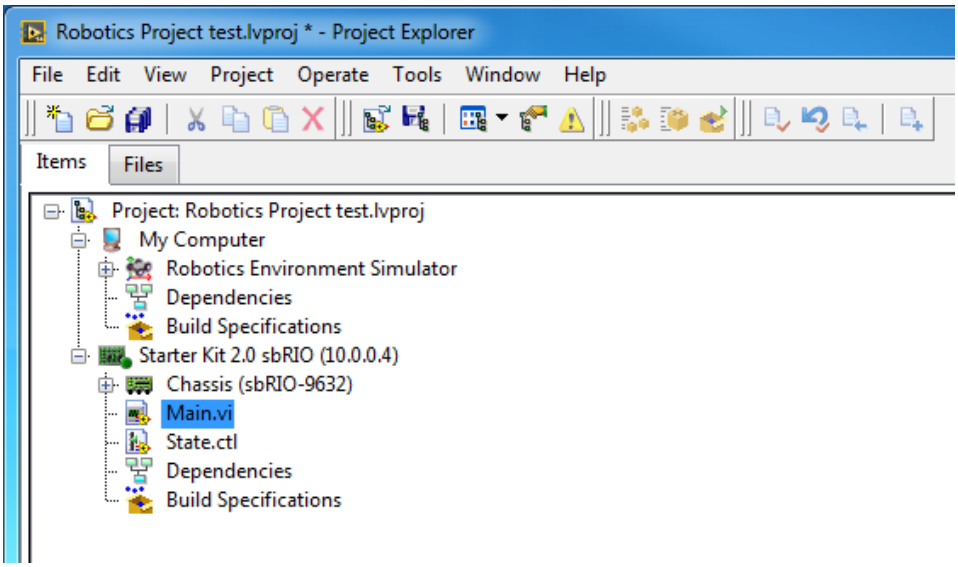


Figure 3.33 project explorer window

- Figure 3.34 is the front panel of the project also known as the user interface. Basic controller and monitors are placed in this panel to help the user operate and comprehend how the starter kit works.
- Note that the start button has to be pressed in order for this program to start working. The forward velocity controls whether the robot moves forward when it

is adjusted to a value greater than zero, and move backward when the value is less than zero.

- The angular velocity on the other end controls the rotation movement of the robot. When adjusted to a value greater than zero, the robot will turn or rotate from left to right. And when to a value less than zero, it will turn or rotate from right to left.
- To view the block diagram of the project, click on **window** in the toolbar and then select **view block diagram**. A new window should open as shown in Figure 3.35.

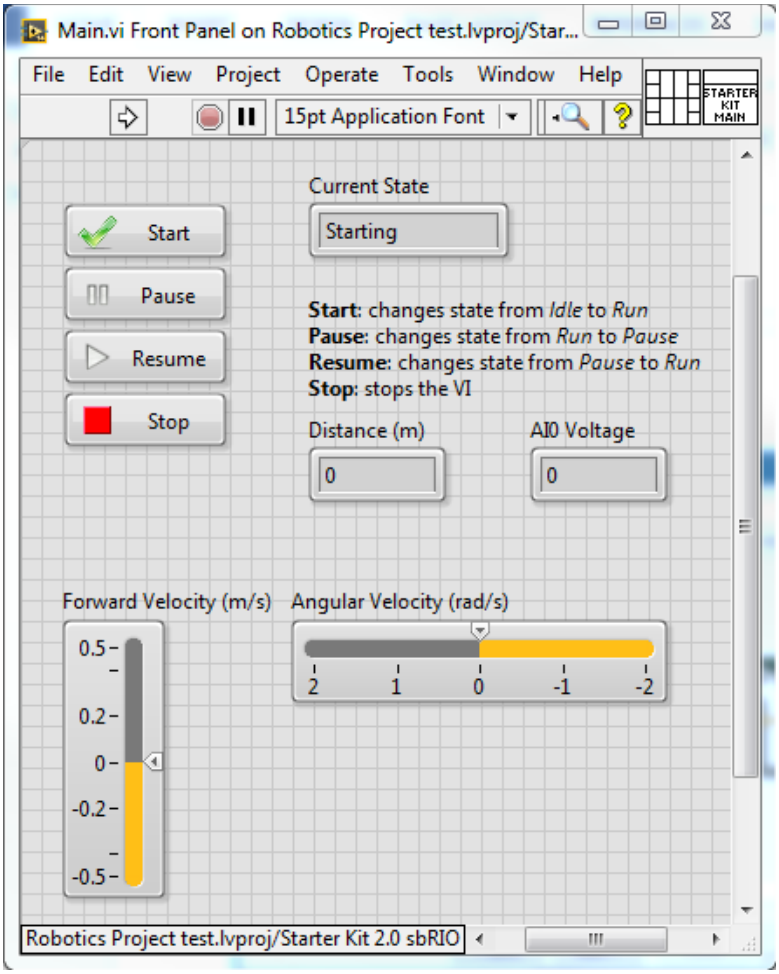


Figure 3.34 project's front panel

- The block diagram in Figure 3.35 shows all the elements and connection used to make the project function properly. The aim of this section is to be able to control the movement of the robot. Therefore, we only focus on the part of the block diagram that deals with the movement as shown in Figure 3.36.

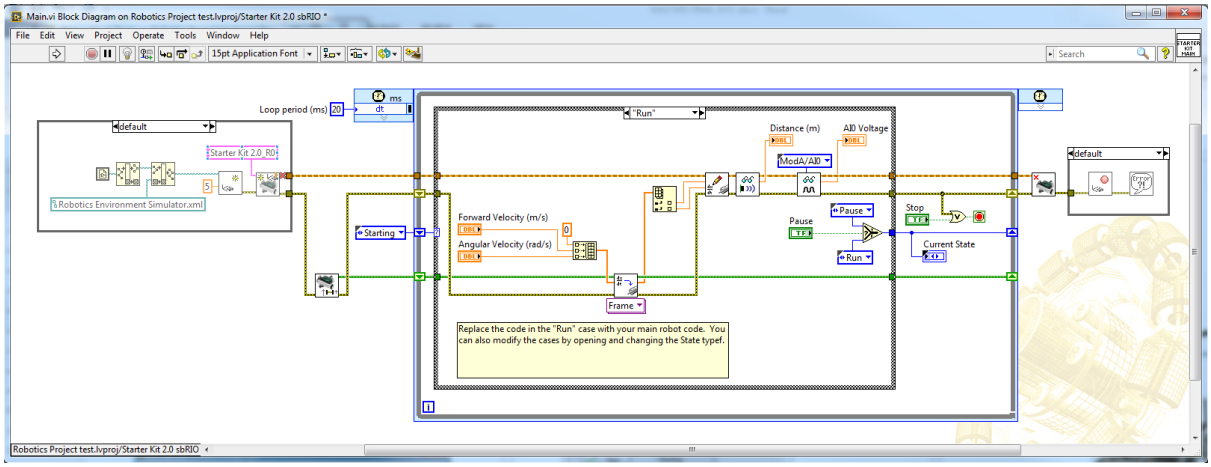


Figure 3.35 project's block diagram

- The **forward velocity** and **angular velocity** (referring to Figure 3.36) are our main focus. In the final code that controls the AGV, they are changed into variables that can be accessed from any part of the final block diagram. This allows the AGV to perform all the actions mentioned in Table 3.6.

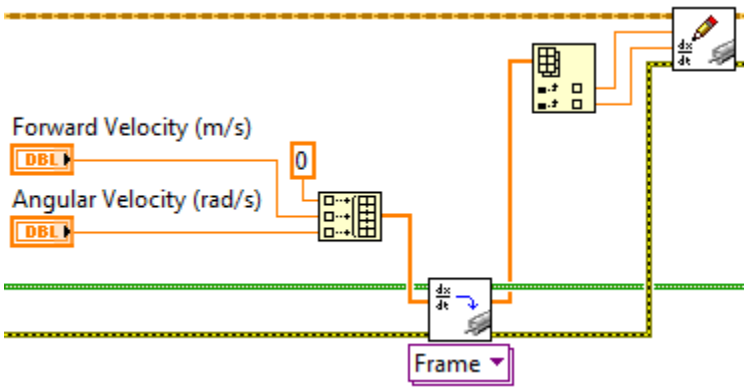


Figure 3.36 Motors control block diagram

After running a few tests, I have come to the conclusion that the optimum velocity for the AGV to perform all of its action are as shown in Table 3.7 below.

Table 3.7 Optimum Velocity

Action	Forward velocity	Angular velocity
Stop	0	0
Move forward	3	0
Move backward	-3	0
Turn right	0	1
Turn left	0	-1

3.3.3.2 Step 2: Decision

The AGV is placed into an environment as shown in Figure 3.37, where it is required to follow the black line on a white surface from one point to the other without getting lost while do it.

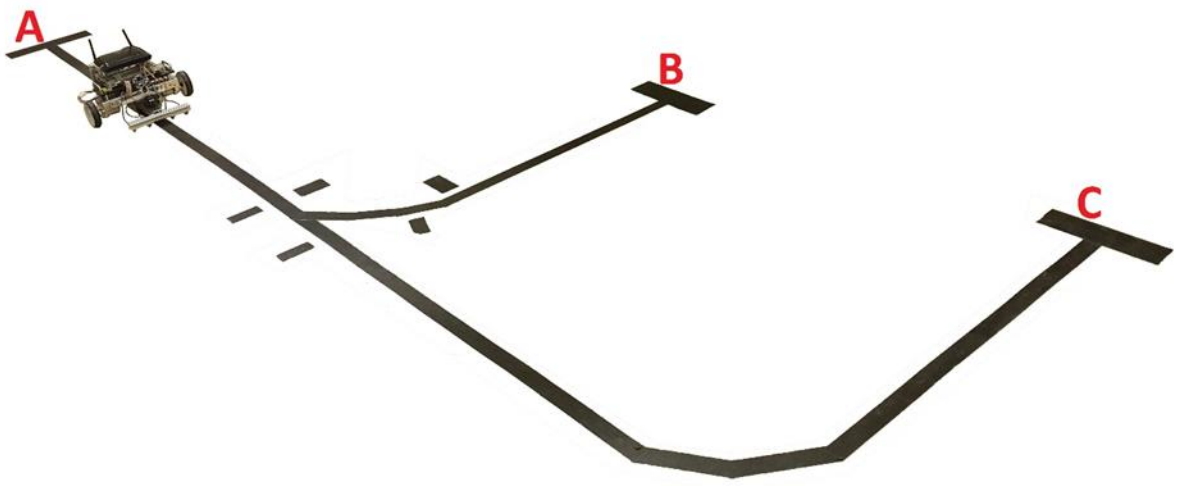


Figure 3.37 AGV in its environment

From Figure 3.37 above the AGV is set to follow one of the following paths:

- From A to B
- From A to C
- From B to A
- From C to A

Note that there is no path set for the AGV to go form B to C or from C to B because each point represents the location of components of the entire system.

- Point A: is the location of the storeroom
- Point B: is the location of the first Assembly line
- Point C: is the location of the second Assembly line

With that in mind, there is no need for the AGV to travel from one assembly line to the other.

The decision process of the AGV mostly depends on the inputs of the sensors that are used to follow the line. The commands form the assembly lines just tells the AGV where it has to go.

From Figure 3.38 below, the AGV is placed on the line with two sensors on the left of the line and two sensors on the right of the line. Table 3.4 explains the role of each sensor.

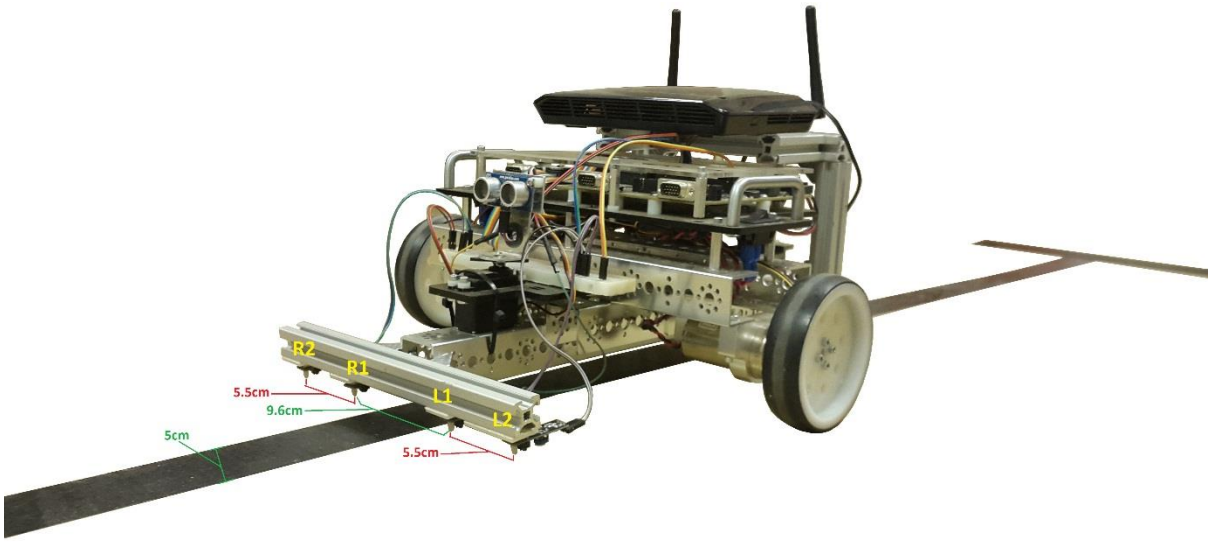


Figure 3.38 AGV on the line

Figure 3.38 highlights the location of the sensors compared to each other and the size the line that the AGV follows.

To follow the line, the AGV makes use of only two sensors (R1 and L1). These sensors have a default value of 1 and changes to 0 when the sensor passes over the black line.

Table 3.8 Decisions and Actions based on sensors input

R1 value	***L1 value***	***Decision and action***
0	***0***	***Stop movement***
0	***1***	***Turn right***
1	***0***	***Turn left***
1	***1***	***Move forward***

The other two sensors are used as check point counter. The check points next to the black line are utilized by the AGV to determine its location compared to where it is going.

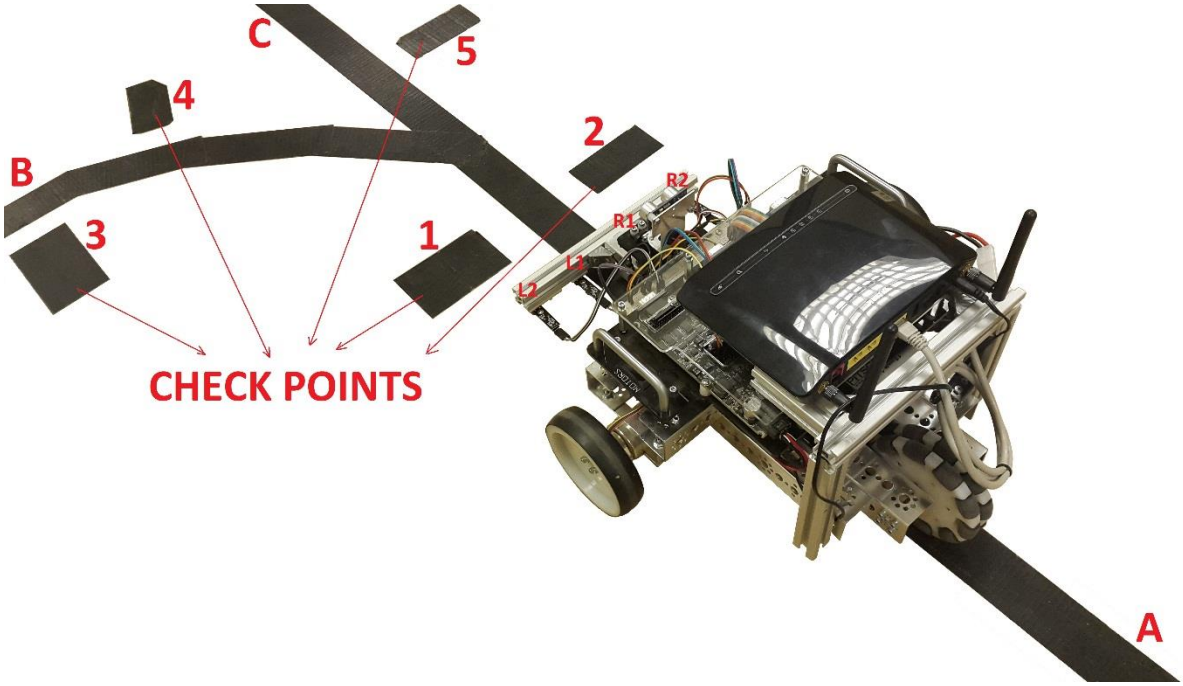


Figure 3.39 AGV at the cross line

Figure 3.39 clearly highlights the 5 different check points that are next to the black line. Now, based on the information obtained from Figure 3.37, the AGV has 4 different paths and if we combine that with the information in Table 3.1 we get the decision and actions table shown below in Table 3.9.

Table 3.9 Decision and actions based on Assembly lines Commands and sensors

Assembly line's commands	Description	Path to follow	Check points in Figure 3.39
0	Stop	N/A	N/A
1	Go to Assembly line 1 location	A to B	1 and 3
2	Go to Assembly line 2 location	A to C	2 and 5
3	Go back to storeroom from assembly line 1	B to A	3 and 1
4	Go back to storeroom from assembly line 2	C to A	5 and 2

Table 3.9 is used a base reference for the programming of the AGV. The AGV program is a case structured program in which each command from the assembly line triggers a specific case that needs to be executed.

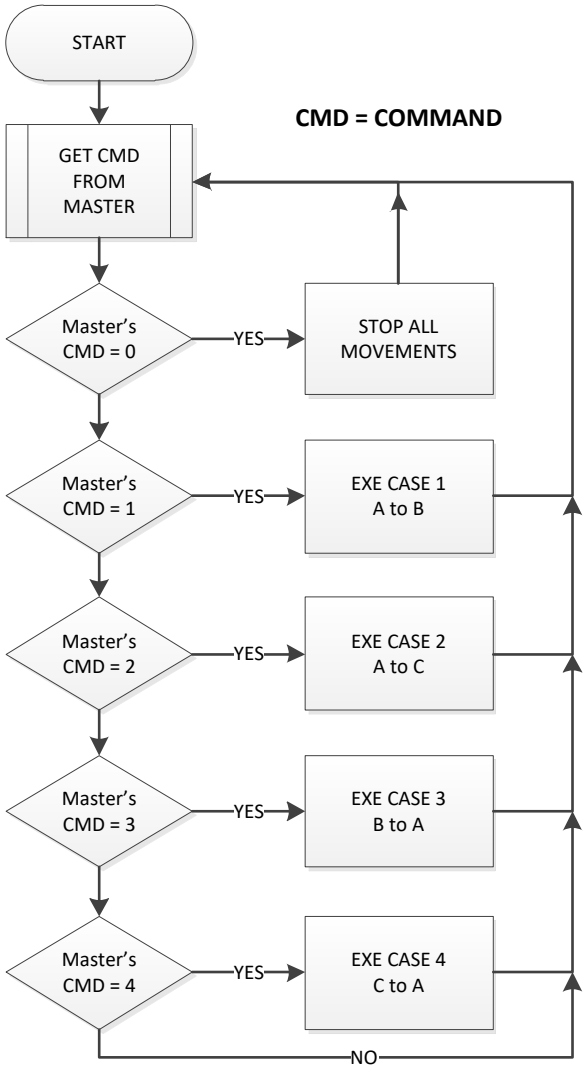


Figure 3.40 AGV program flowchart

Each one of the cases shown in Figure 3.40 above is a subroutine of its own that controls the AGV allowing it to follow a specific path. For example: in CASE 1, the AGV will follow the Path from A to B.

Table 3.10 AGV program reference full table

	Assembly line cmd	description	the path to follow	check point count	sensor state		Action
					R1	L1	
	0	stop	N/A	N/A	X	X	stop
	1	Go to M1	A to B	0	1	1	drive forward

CASE 1					1	0	turn left
					0	1	turn right
					0	0	drive forward
				1	X	0	turn left
					X	1	drive forward
				2			drive forward
					1	1	
					1	0	turn left
					0	1	turn right
					0	0	drive forward
				3	X	X	turn left
				4	X	0	turn left
					X	1	stop
CASE 2	2	Go to M2	A to C	0			drive forward
					1	1	
					1	0	turn left
					0	1	turn right
					0	0	drive forward
				1	0	X	turn right
					1	X	drive forward
				2			drive forward
					1	1	
					1	0	turn left
					0	1	turn right
					0	0	drive forward
				3	X	X	turn left
				4	X	0	turn left
					X	1	stop
CASE 3	3	Go to Storeroom from M1	B to A	0			drive forward
					1	1	
					1	0	turn left
					0	1	turn right
					0	0	drive forward
				1	0	X	turn right
					1	X	drive forward
				2	0	X	turn right
					1	X	drive forward
				3			drive forward
					1	1	
					1	0	turn left
					0	1	turn right
					0	0	drive forward

				4	X	X	turn left
				5	X	0	turn left
					X	1	stop
CASE 4	4	Go to Storeroom from M1	C to A	0	1	1	drive forward
					1	0	turn left
					0	1	turn right
					0	0	drive forward
				1	X	0	turn left
					X	1	drive forward
				2	1	1	drive forward
					1	0	turn left
					0	1	turn right
					0	0	drive forward
				3	X	X	turn left
				4	X	0	turn left
					X	1	stop

Table 3.10 represent the way that the AGV program functions. It includes all the essential details of the program's operation with the exception of the left sensor 2 (L2) and right sensor 2 (R2). Those two sensors as mentioned in the previous pages of this document, are used to count the check points next to the line that is being followed.

Referring back to Figure 3.39, it has been shown that there are 5 different check points. But, if we consider the case in which the AGV follow the path from A to C and using L2 to count the check points, we will notice that there will be a point where L2 will cross the line that goes to B and will count it as a check point as well.

The actions perform by the AGV differs at different check points, for example: in case 1, the AGV must go from A to B. At the start, the check point count is equal to zero. Therefore, the AGV uses the sensors R1 and L1 to follow the line

- From the moment that L2 reads the first check point, the check point count will be incremented by 1 and will do so at every single check on it path.
- When the counter is equal to 1, the AGV knows that it is at the point where the path to B and C crosses and it has to keep left to go to B. Therefore, all the data coming from the sensors at the right side of the line are ignored.
- After keeping left, the AGV will come across another check point and the counter will increment again bringing its value to 2. At this stage the AGV knows that it is on its way to B and it will follow the line using both sensors (R1 and L1).

- At each destination point of the line (A, B, and C), there is a T shape that the AGV uses to know that it has reached its destination. When the AGV reads the T at any of these points (A, B, and C), it will rotate to the left and reposition itself to the line. This happens every time when the check point count reaches a value of 3 and 4 in case 1, 2, and 4, for case 3 it does it for the values of 4 and 5.
- Every time that the AGV arrives at its destination and finishes to reposition itself on the line, it stops all motions, sends a message to the assembly lines, and wait for further commands to execute.

Figure 3.41 below is the flowchart of Case 1, but the same principle and codes are applied to all the other cases in the AGV control program.

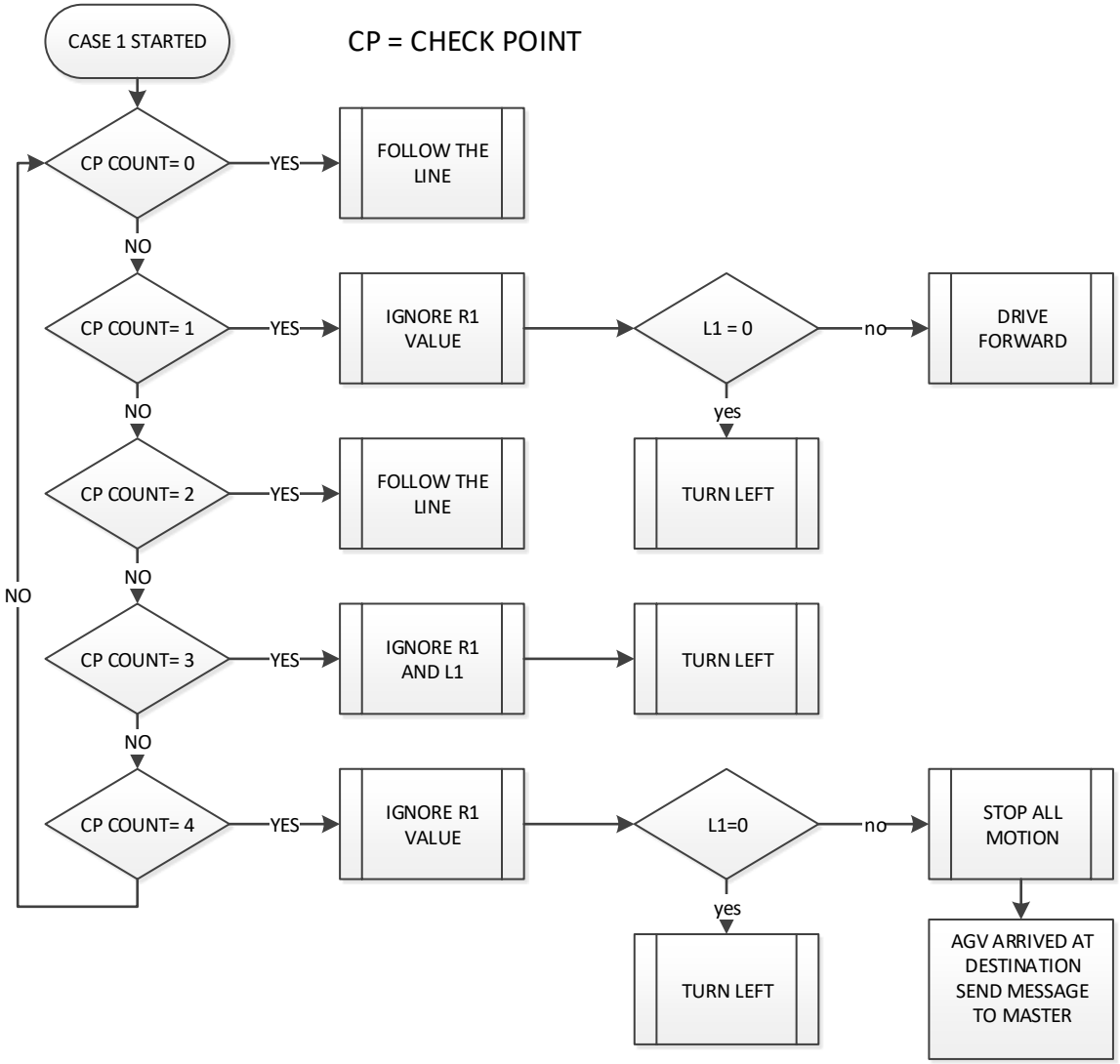


Figure 3.41 Case 1 flowchart

3.4 The storeroom

The storeroom operation mainly consists of three separate processes are the motion, identification and the process control. This section covers each of the processes and gives more detail on the work done in order to complete this section of the project. [9]

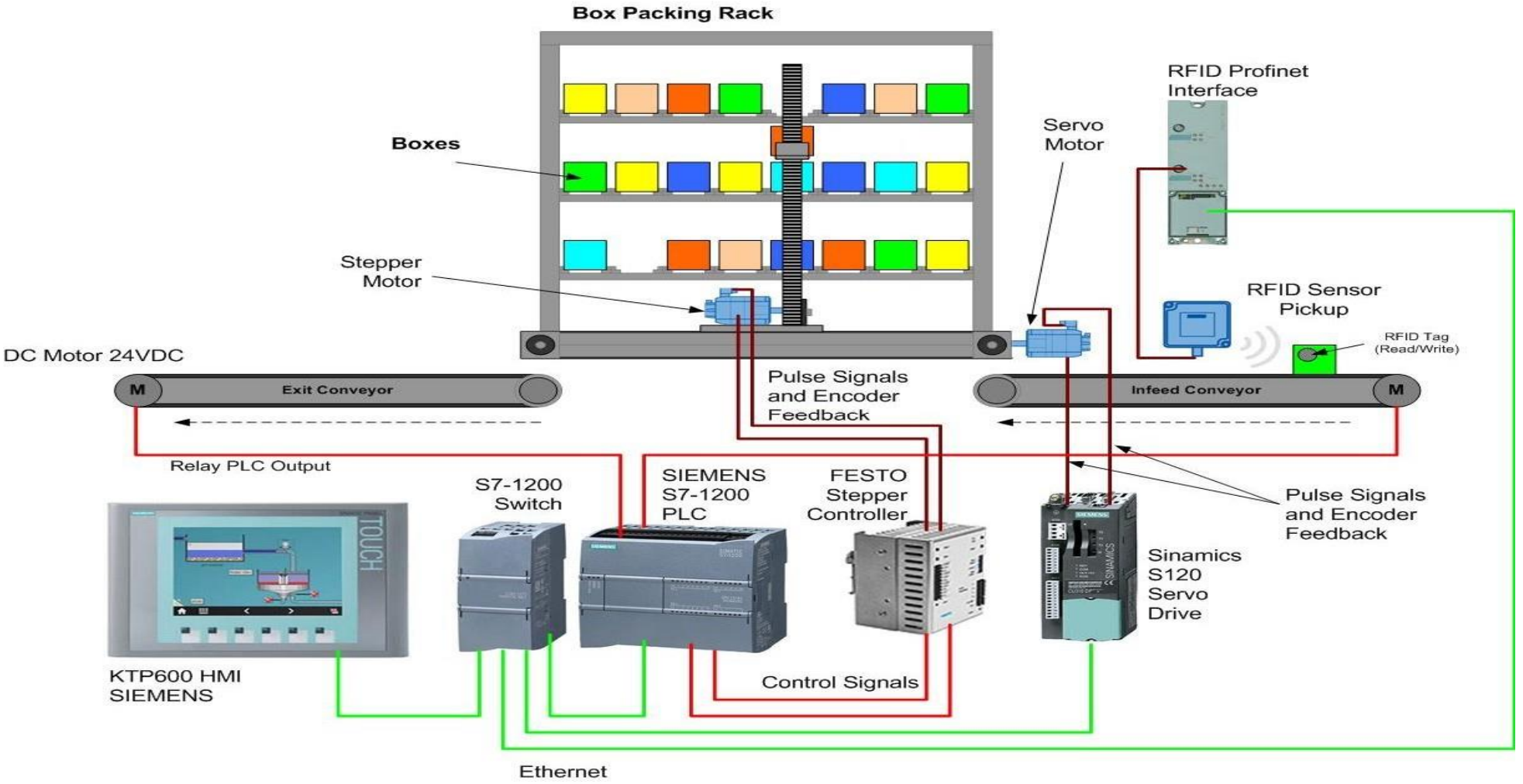


Figure 3.42 Storeroom system connection diagram

The above Figure 3.42 shows the storeroom system's connection diagram. Its main component is the PLC, and it serves as the brain of the entire system by monitoring its input and updating its output according to the system's requirement. The PLC is programmed to control each and every process that happens in the storeroom

3.4.1 Main process (PLC Main program)

The main function of the PLC is to monitor the execution of the following storeroom procedures:

- Parts/products identification
- Storage of Parts/products
- Retrieval of Parts/product from storage

The simple operation of the storeroom is as followed:

At the start of the storeroom system, PLC waits for the input signal that indicates that a product is ready for pickup at the input conveyor belt. At the moment that the signal is received, the PLC uses RFID scanner to identify the product, determine the specific location where it has to be stored and finally uses the stepper motor to move to the storage location and store the product.

The Figure 3.43 below gives a more detailed overview of the main program including when an order is placed to the system.

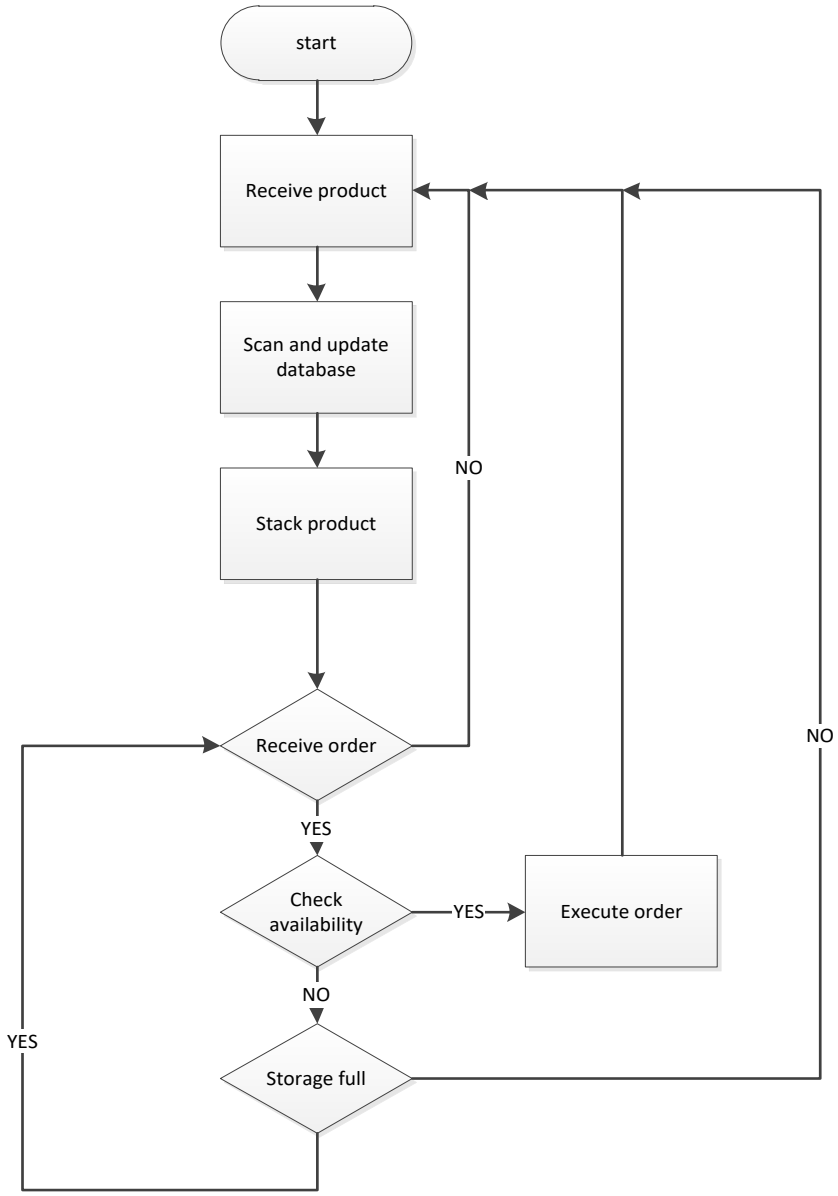


Figure 3.43 Main Storeroom's PLC program flow chart

Figure 3.43 above is the main storeroom's PLC program flowchart. When started, it receives products, and then scans them to update the products database, and finally stacks the received product in the storage area. Afterward, if an order is received, the system will check for availability before executing the order. And if the storage is full, the system will constantly loop to wait for orders to be received.

From the main program flow chart, one can divide the program into three sub-processes that will simplify the work load and improve troubleshooting of the code.

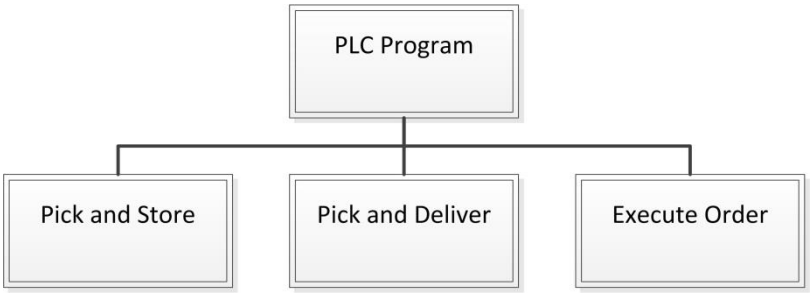


Figure 3.44 PLC program break down structure

Figure 3.44 shows the breakdown of the PLC program. It shows the three parts namely: Pick and Store, Pick and Deliver, and Execute Order. These parts are further explained in the headlines below.

3.4.1.1 Pick and store

This is the part of the program that is in charge of the pick and store. Its basics operations are as followed:

- It starts when a product is ready for storage at the input conveyor, the PLC receives and process the input signal by retrieving the coordinate of the input conveyor from its memory then sends it to the stepper motor drives. The stepper motors controlled by the will then execute the motion and get ready for pick up.
- After the motion has been completed, the PLC receives a signal from the stepper motor drives that lets it know that the motion is completed and the forklift is at the pickup point. The product will then be picked up from the conveyor and the signal will again be sent to the PLC to require for the storage place coordinate.
- The PLC will get once again the coordinate of the storage place from it memory and send it to the drive for the motion execution.
- After the motion has been executed, the forklift will place and store the product at the designated location provided by the PLC.

NB: The PLC has a list of designated area for each type of the goods / parts that are stored in the storeroom. This makes it efficient for order execution and facilitates the work of the storeroom manager to know the exact location of a specific product in the entire storeroom and the specific amount of the goods / parts in stored.

Each product that comes for storage has an RFID tag on it that is specific to the type of that product. The PLC, with the help of the RFID scanner, determines the storage location of product using the information read on its RFID tag.

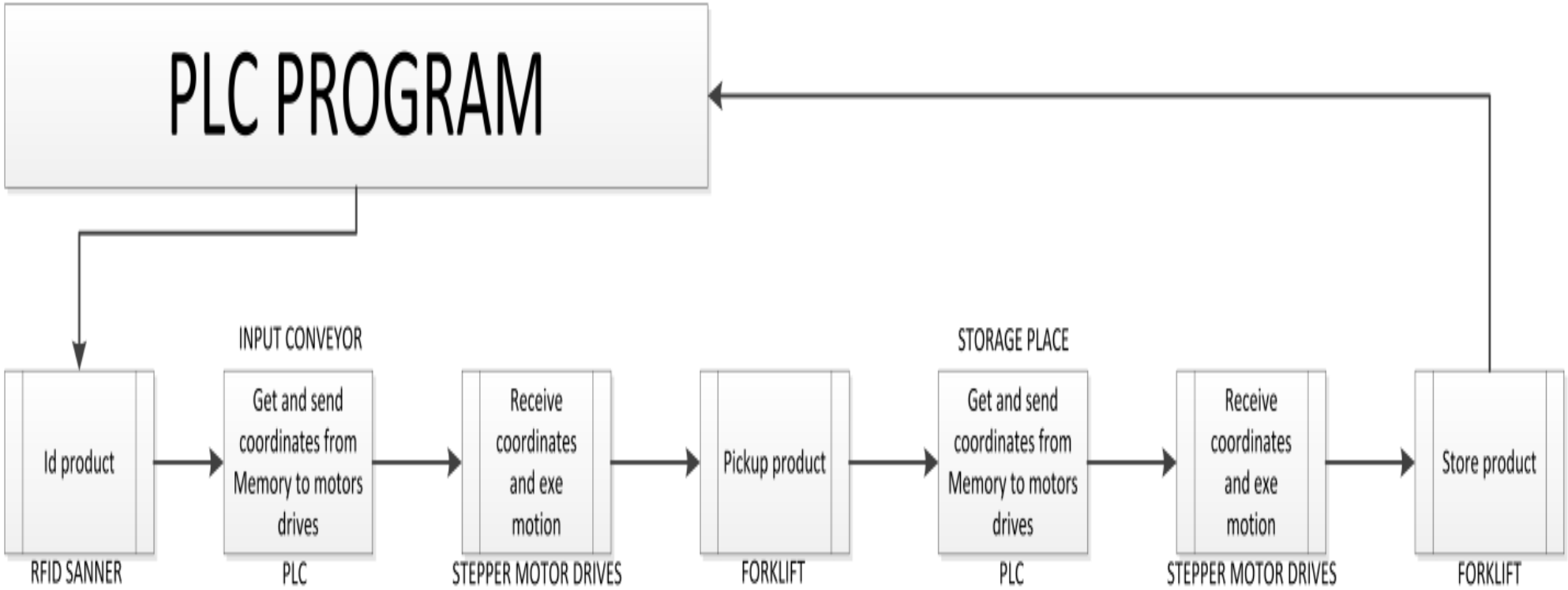


Figure 3.45 Pick and store process flow

3.4.1.2 Pick and deliver

This part of the program is an exact replicate of the pick and store except for the fact that the PLC starts by sending the location of the product that needs to be delivered first, then the location of the exit conveyor. All the motions are controlled by the drives and executed by the stepper motors. Refer to Figure 3.46 below for more details.

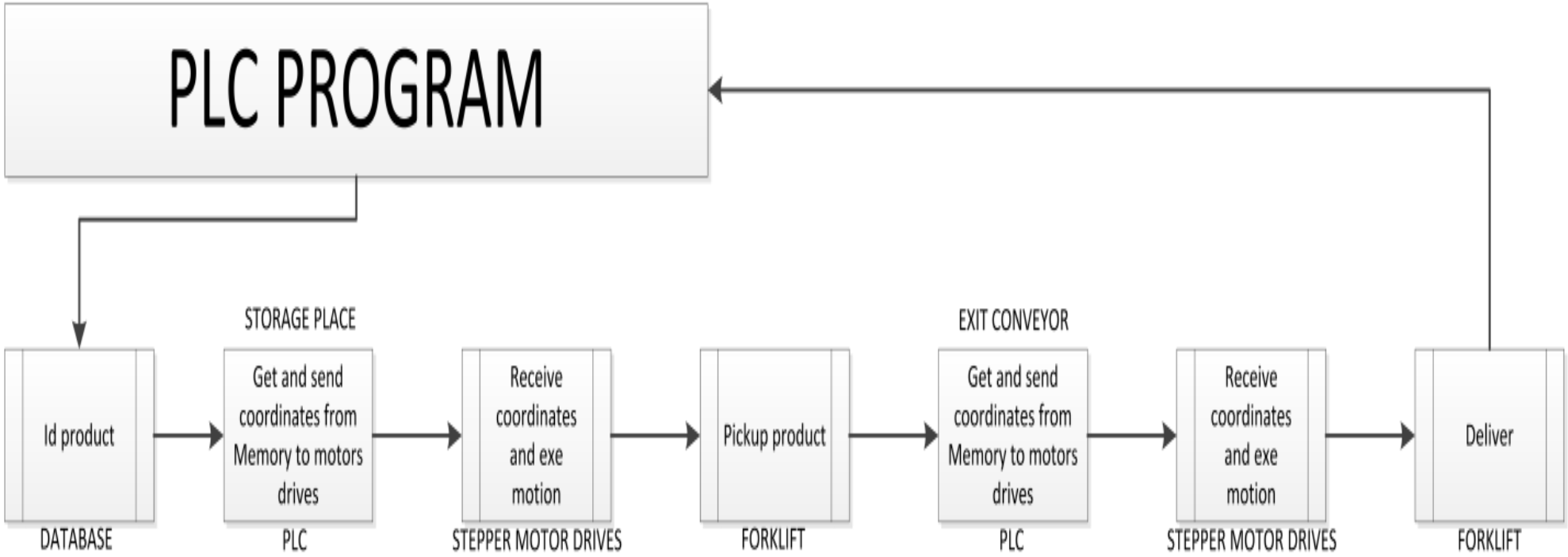


Figure 3.46 Pick and deliver process flow

3.4.1.3 Execute Order

The execution order of program mostly makes use of the pick and deliver portion of the PLC program. When an order is placed for a certain amount of parts, it will first check if all the required parts are available. If not, the order will remain pending and the PLC will continue its process until the required amount is met. The process will then be interrupted for the order to be executed.

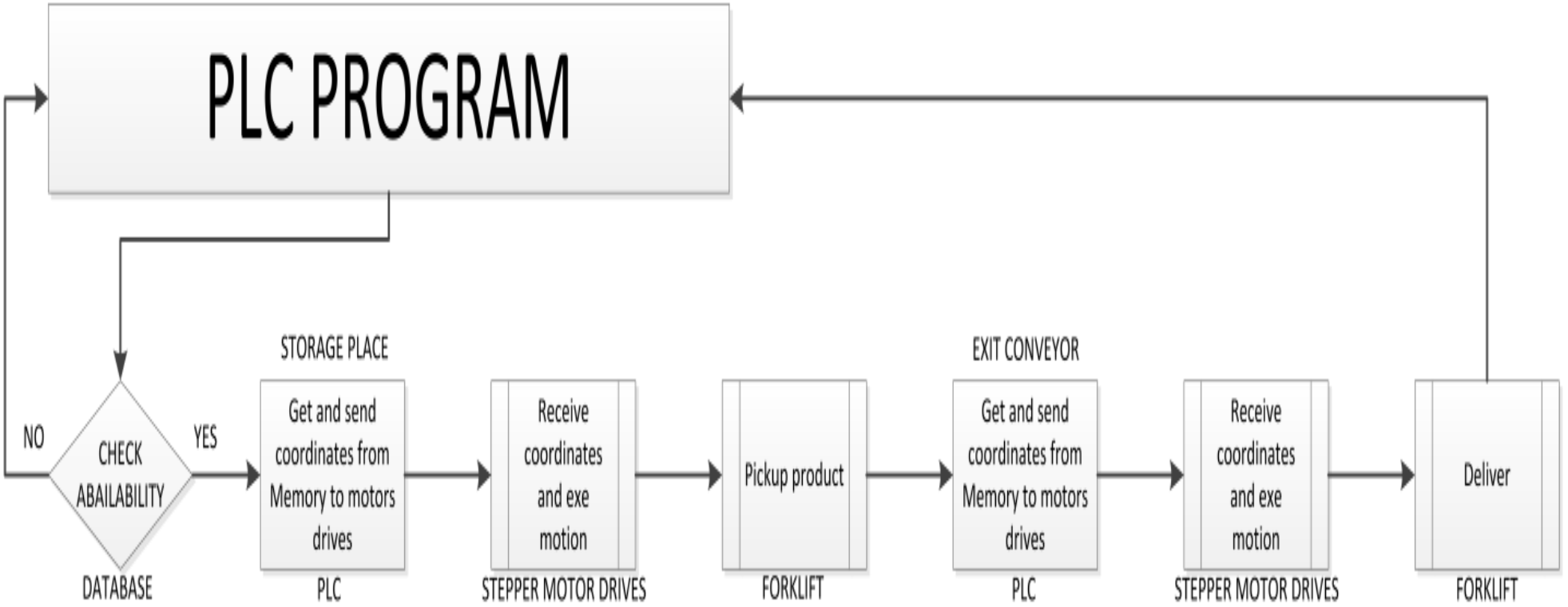


Figure 3.47 Execute order process flow

The whole system is monitor by a human machine interface (HIM) that serves to get the input (placing an order…) from the user and displays the output (number of products/parts currently in the storeroom…) for the user to monitor. The order for parts collection can also be placed over the network link between the storeroom and the Assembly lines. This network link allows for constant communication between the assembly line and the storeroom

4 TESTING AND RESULTS

This chapter reports on the sequence of tests that were carried out to verify the operation of the system and its subcomponents.

The tests are designed to verify the vital system's functions first, and then verify system sub-processes, and finally testing the system as a unit. The tests and results have been split into the next section:

- The assembly line: this section is dedicated to testing the main function of the assembly line and analyze the results obtain after testing.
- The worker: in this segment, the tests and results of the AGV are discussed.
- The storeroom: the results and the tests for the storeroom are discussed in the section.
- The system: the overall system's tests and results are analyzed and discussed in this chapter section.

4.1 The assembly line

The assembly line is an assembly line computer simulation as explained in literature review chapter 2 (referring to 2.1 Assembly lines). It has been programmed to perform information sharing, task management, and resource usage negotiation as described in the methodology chapter 3.

This part of chapter is dedicated to the performance tests that will check the function of the assembly lines and the evaluation of the obtained results.

4.1.1 Test

The tests in this section are meant to test the basic functionality of the assembly line. The assembly line should be able to create or connect to a server, share the task list, and negotiate on resources usage.

The test done bellow in 4.1.1.1 is made to test the communication and the task priority negotiations.

4.1.1.1 Communication and task priority negotiations

This test is run to determine whether the assembly lines can successfully establish and maintain a commutation line in which they will share information about the list of tasks that they will perform. They will also use the network to share constantly the status of the tasks that are performed and negotiate on resource management.

Figure 4.1 Assembly line program User Interface

Figure 4.1 above is the user interface of the assembly line's program. It allows the user to see what is currently happening in the system. The user interface is divided into 3 different areas as followed:

- Area 1: this area is reserved for the messages shared between the assembly lines. It allows the user to see the progress of the operation being performed by either one of the assembly lines.
- Area 2: is used to display the list of assembly lines or Assembly line that are currently connected to the system.
- Area 3: this part of the user interface shows the list of tasks that have to be performed by the assembly lines. It shows the task list of whatever assembly line selected in the second user interface area.

4.1.1.1.1 Test setup

Using Figure 4.1 as reference, the test is set in the following manner:

- The lists of tasks are loaded on the computers that will serve as assembly lines
- Start the first assembly line
- Start the server
- Connect the first assembly line to the server
- Start the second assembly line
- Connect the second assembly line to the server

After the second assembly line is connected, the rest will run automatically until the point where the AGV needs to be connected to the server for the process to continue. The results of this test are discussed below in the next section.

4.1.2 Results

These are the results obtained from the tests done in the previous section. They have been divided into sub-sections that are as follows:

- The result obtained after starting the server
- The result of the connection to the server
- And the communication and basic operation of the assembly lines

4.1.2.1 Starting sever

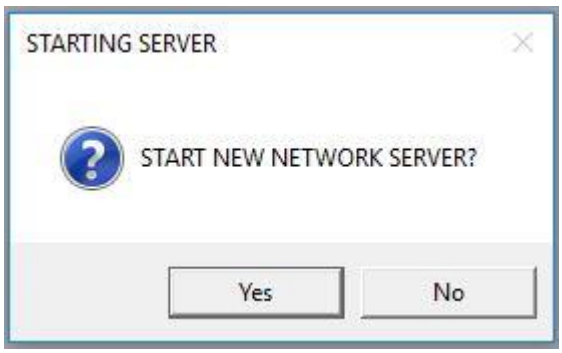


Figure 4.2 Start server message box

The assembly line is mainly a client server application that runs on the same application rather than two different apps (one app for the server and one for the client). Only one server can operate at the time. Therefore, this message box purpose is to allow the user to choose whether to start the server on this assembly line or to connect to an already running server.

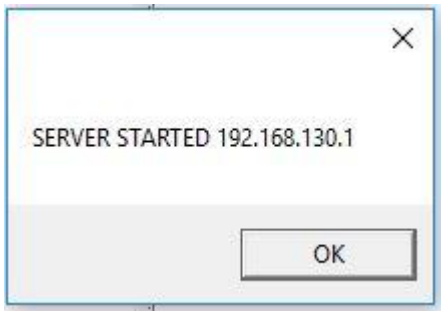


Figure 4.3 Server's IP message box

When the user chooses to start a new server, the following message box shown in Figure 4.3 will appear as a result of a successful start of the server. It shows the IP address of the server that the user will use to connect another client to this server.

4.1.2.2 Connecting to server

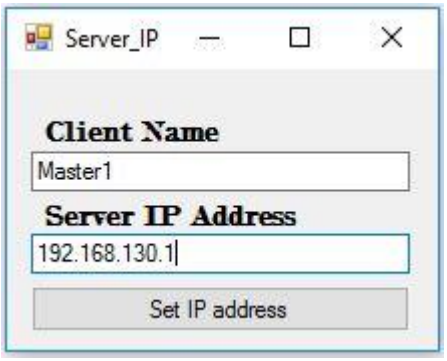


Figure 4.4 Server connection user interface

The user interface in Figure 4.4 is used to set the client's connection to the server. The main settings are the client's name which in this case is **Assembly line1** and the server's IP address.

Note that each client is considered as an assembly line on its own, so therefore, Assembly line1 is the name of the first assembly line and assembly line2 will be the second assembly line name. The IP address will remain the same because they all have to be connected to the same server in order for the system to work.

4.1.2.3 Assembly lines communication and basic operation results

Figure 4.5 illustrates the operation test of the two assembly lines side by side. It clearly shows the different steps take in the information sharing and resources management process taken by the assembly line

Six results are shown in Figure 4.5 below in area 1 of the user interface (refer to Figure 4.1) are the following:

- Result 1: it is a message received from the server indicating that assembly lines have been successfully connected to the server and are ready to share messages.
- Result 2: this message is displayed after the list of tasks from the other assembly line has been received and successfully saved to the current assembly line's program to be used later on as a reference. The list received can be seen by selecting the assembly line where it is from on the second area of the user interface. Refer to Figure 4.1 above on page 92
- Result 3: this is a message from the other assembly line that serves as a notification that a task is about to be started.
- Result 4: is the name of the task that is about to start. The program uses this information to compare the priority level to its own task and determine whether it has priority over the resources usage.

- Result 5: this is a message shared between the assembly lines as a result of the priority level comparison process. Refer to section 4.4.2 of the result on page 109 for more details.
- Result 6: these messages between the assembly lines shows the results of the negotiations where one assembly line takes control of the resources to execute its task and the other waits for the assembly line with highest priority task to finish with the utilization of the resources. The assembly line with the highest task priority will wait for the AGV to be online before requesting for parts from the storeroom.

Area 2 of the user interface shows the list of assembly lines currently connected to the server. When an item is selected in the list shown in area 2, area 3 shows the task list of the selected item. Refer to Figure 4.5.

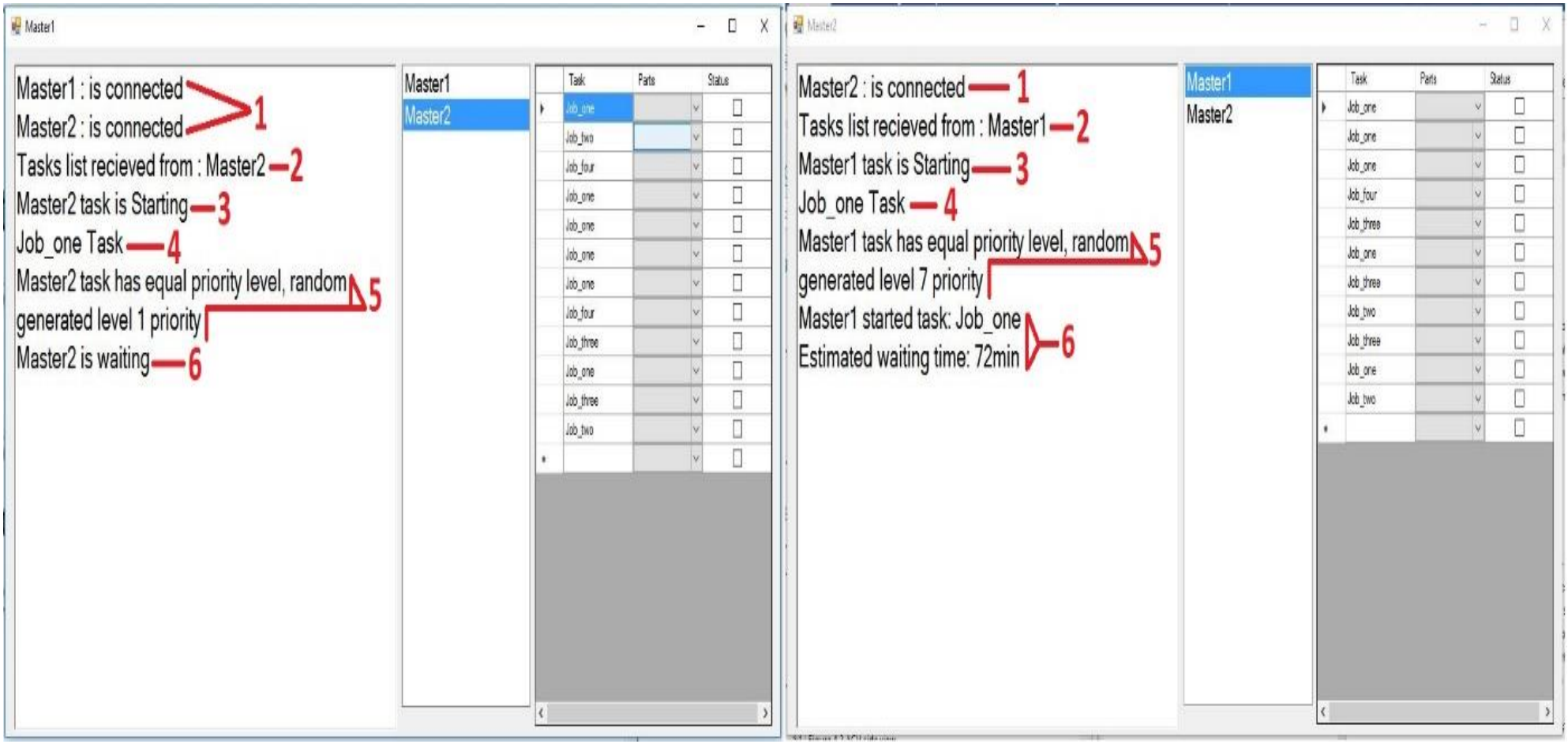


Figure 4.5 Assembly lines communication

Figure 4.5 shows a side by side view of the operation of the two assembly lines. Its functions are explained in the paragraphs preceding it.

4.2 The worker

The worker is an AGV as described in chapter 1. Figure 1.1 in the introduction chapter, shows that the AGV is the link between the assembly lines and the storeroom. It is programmed to follow a black line on a white surface as explained in the third chapter. This segment of this chapter focused on the test that will determine whether the AGV operates correctly. The author also describes and analyses the results obtained are each test.

4.2.1 Test

These tests are setup to test the main functions of the worker. The worker should be able to effectively follow the line from the storeroom to the assembly lines and back. It should receive commands over the network from the assembly lines and execute them accordingly. The coming paragraphs explain how those test are set up.

4.2.1.1 Line following test setup

The AGV, in this test, is placed in an environment where it has to follow a black line on a white surface. The purpose of this test is to check the line following code that is loaded on the AGV's motherboard to see if it will be able to correctly see and follow the line. The test is conducted as following:

- Place the AGV on the line with the sensors on either side on the line.
- Turn on the assembly line switch Then wait about 10 to 30 seconds for the program in the memory to initiate.
- Turn on the motors switch to start running the AGV.

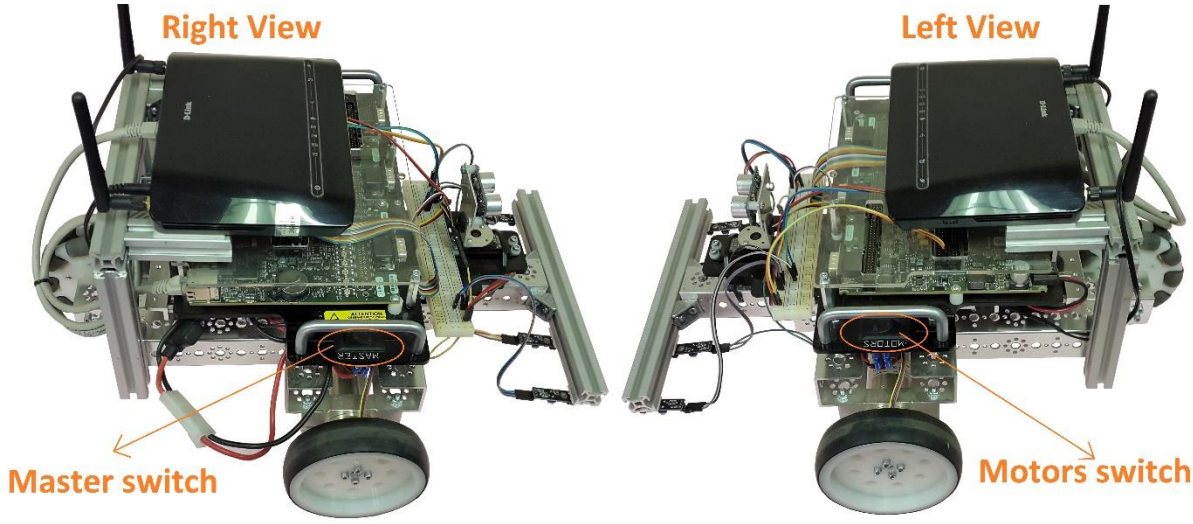


Figure 4.6 AGV side view

Figure 4.6 above is the worker's side by side view. It serves as reference for the location of the switches that control the LabVIEW robotic starter kit.

This test will also determine the optimum speed at which the AGV can properly follow the line without going off track.

4.2.1.2 Communication and network command

This test is designed to check the communication between the AGV and the computer. A simple application runs on the PC to send commands to the AGV to tell it where to go. With each command received, the AGV performances a specific section of its control code. This allows the author to not only to test the network communication but also the AGV's destinations repertory. Referring back to Table 3.1 on page 61.

4.2.2 Results

The following results obtained from the test ran in the previews section. The author first discusses the AGV's line following results then the communication and network commands results.

4.2.2.1 Line following

The result of the line following test was as expected, the AGV followed the line successfully. The accuracy of the line following however depends on the speed at which the AGV is traveling. Referring to the calculations bellow, the maximum velocity of the AGV was determined.

$$\max DC\ motor\ velocity = 152\ rpm$$

$$\omega = \frac{152}{60} x\ 2\pi = 15.92\ {}^{rad}/_{s}$$

$$r = 5\ cm$$

$$v = \omega . r$$

$$v = 15.92\ x\ 5\ x\ 10^{-2} = 0.79\ {}^{m}/_{s}$$

$where$:

$r\ is\ the\ radius\ of\ the\ wheel$

$\omega\ is\ the\ maximum\ angular\ velocity\ the\ AGV$

$v\ is\ the\ maximum\ linear\ velocity\ of\ the\ AGV$

The AGV control program is set in such a way that we use the linear speed to go forward or backward depending on the sign of the number set to that variable (+ forward and – backward). The angular velocity is used to control the direction in which the AGV turns (+ right and – left).

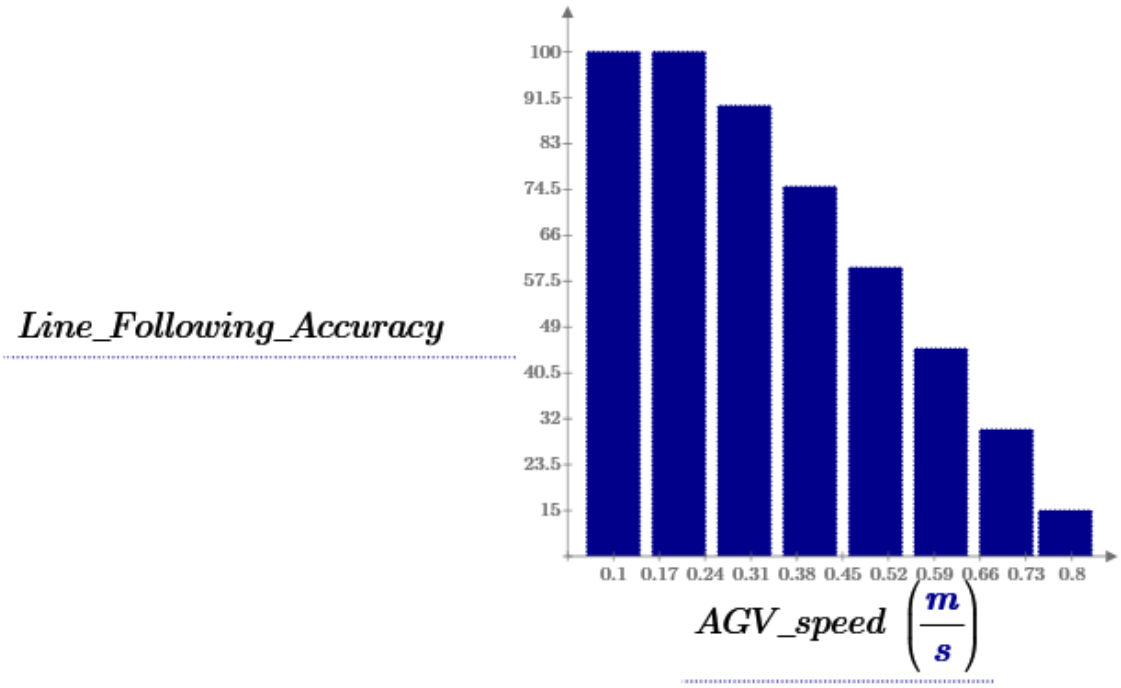


Figure 4.7 Line following speed test results

Figure 4.7 above, shows that the line following accuracy decreases when the AGV speed increases. It also indicates that the best speed setting to an accurate line following ranges between 0.1 to 0.38.

4.2.2.2 Communication and network command

The AGV and PC wireless communication work successfully, the sample program application running on the PC successfully sent messages or commands to the AGV. Figure 4.8 below is the user interface of the application designed to test the communication and commands from the PC to the AGV. Each button represents a command that is sent to the AGV when it is clicked. Refer to Table 4.1 bellow for the description of each button on Figure 4.8 and the results of the reaction on the AGV when they are clicked.

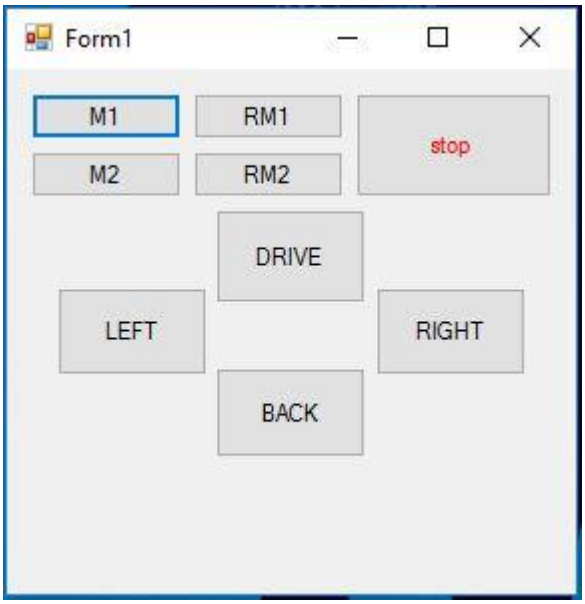


Figure 4.8 Communication and commands testing application

Figure 4.8 is the Communication and commands testing application as explained in the paragraph above. The function and description of each buttons on the user interface are discussed in the table below.

Table 4.1 Application buttons and description

Button Label	DESCRIPTION	AGV Action after message reception referring to Figure 4.9 below
M1	Go to assembly line 1	The AGV received the command and successfully travelled from A to B
M2	Go to assembly line 2	The AGV received the command and successfully travelled from A to C
RM1	Return to storeroom from assembly line 1	The AGV received the command and successfully travelled from B to A
RM2	Return to storeroom from assembly line 2	The AGV received the command and successfully travelled from C to A
stop	Stop motors	The AGV received the command and successfully stopped all motion.
DRIVE	Drive forward ignoring sensors inputs	The AGV received the command and successfully executed it
LEFT	Rotate left ignoring sensors inputs	The AGV received the command and successfully executed it
RIGHT	Rotate right ignoring sensors inputs	The AGV received the command and successfully executed it

BACK	Drive backward ignoring sensors inputs	The AGV received the command and successfully executed it

The Table 4.1 above describes the functions and commands generated by the buttons in the user interface shown in Figure 4.8. This table summarises the user interface, describes the commands sent to the AGV by each button when clicked, and the results obtain when the AGV receives the command. It is furthermore used to test the communication and the response of the AGV and the command application. As results, the third column of the above table show that the results of the tests were successful for each of the buttons shown is Figure 4.8.

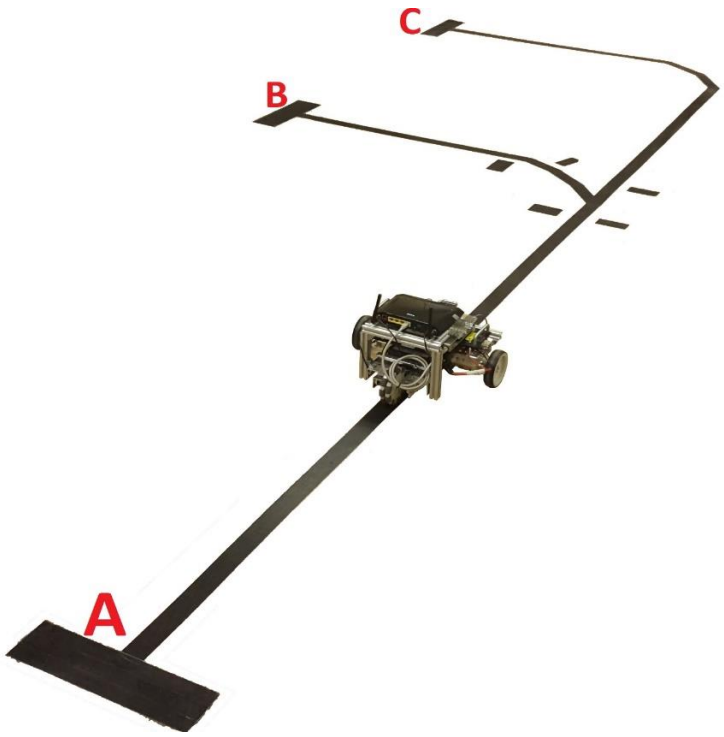


Figure 4.9 Line following AGV

Figure 4.9 above, shows the AGV following a black line on a white surface as explained and designed in chapter 2 and 3 respectively. It has three distinctive location labelled A, B, and C. Location A represents the storeroom, B is the first assembly line, and C is the second assembly line's location.

4.3 The storeroom

The storeroom is an automated storage and retrieval system and shown in Figure 1.1 of the introduction chapter and explained in the literature review in chapter 2. It has been built and programmed to perform its main tasks as explained in the methodology in chapter 3.

This section of this chapter discusses the operation and result of the storeroom.

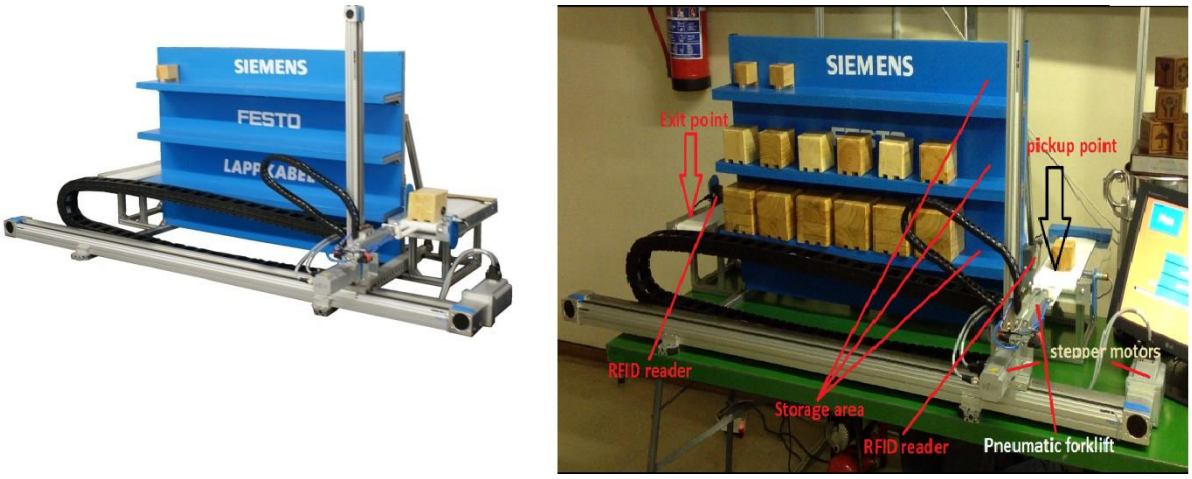


Figure 4.10 Automated storage and retrieval system

Figure 4.10 above is the final is physical built of the storage and retrieval system that is used in the storeroom of this study. The Table 1.1 in the introduction chapter shows that each task is composed of four parts collected from the storeroom. To implement that function, the author used the following:

- Part 1 is represented as a small box
- Part 2 is the medium box
- Part 3 is the big box and
- Part 4 is the combination of a small and big box.

4.3.1 Operation and results

The system shown in Figure 4.10 works in the following manner:

- Each box is fitted with an RFID tag that the system uses to know its properties (box size: big, medium or small)
- The box received at the entry conveyor belt is scanned by the RFID reader. The PLC uses the data from the RFID reader to allocation a storage location for the scanned box.
- The box is then picked up using the motion combination of the pneumatic forklift and the stepper motors and stored in the PLC's allocated storage place.
- This will repeat until the all the storage system is full.

When the assembly line requests for parts from the storeroom, the storeroom system interrupts the packing process to execute the order placed only if all the parts required are already stocked. If the parts required are not yet stored, the packing process will continue until all the part ordered are in storage then will execute the pending order placed.

Figure 4.11 below is the flowchart of the storeroom system. It shows a step by step detailed operation of the system.

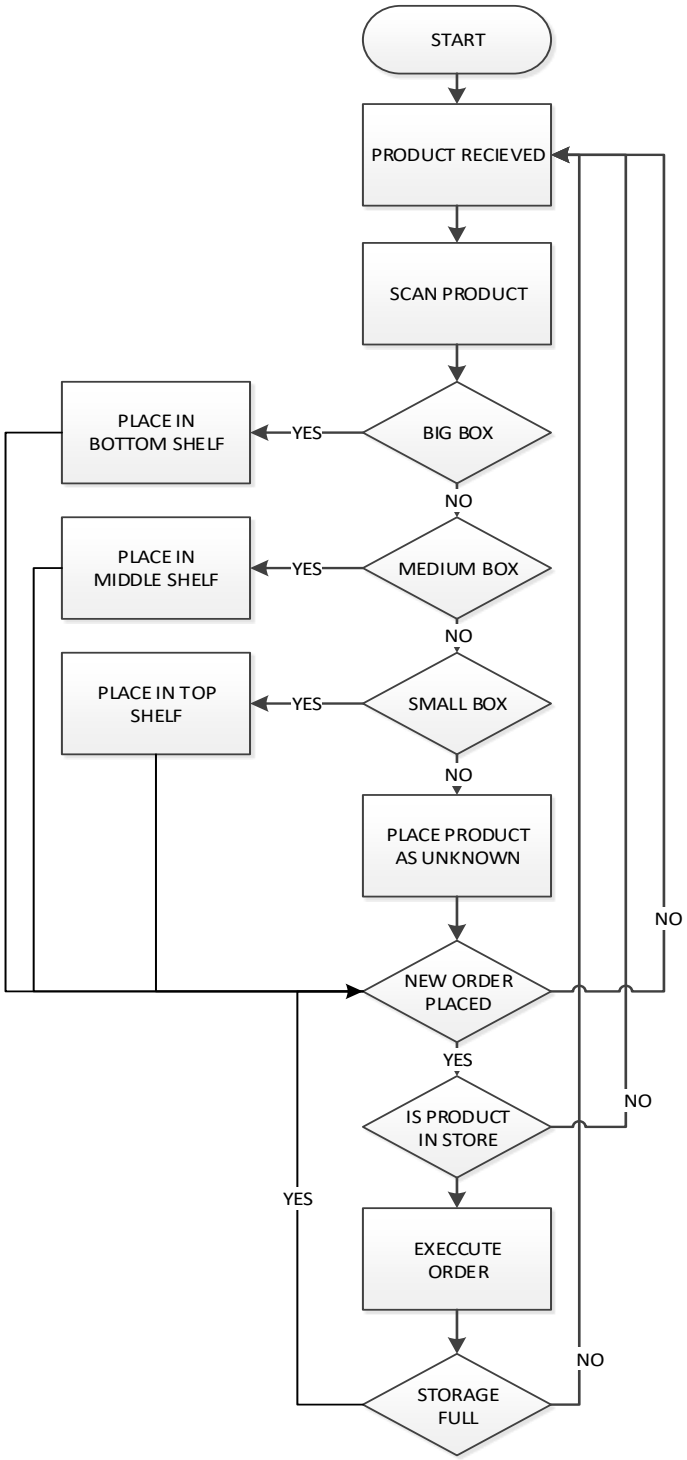


Figure 4.11 storeroom system's operation flowchart

Figure 4.11 is the storeroom system's operation flowchart. After the system starts, the products reception process start, then each product is scan by the RFID scanner. Afterward, the scan results go through a series of condition to determine the location in which the product will be stored. Thereafter, the system will start checking if there are orders for products already store. If there is a pending order, the system will check if all

the products requested are in the storage area to start the execution of that order or else it will continue with the reception of products until the order request is met.

Figure 4.12 Storage allocation

Figure 4.12 is the storage allocation used by the PLC to store and retrieve boxes when they are received or requested by the assembly lines.

Figure 4.13 is the time chart result of the storage process. It shows the time that the system takes to store different boxes. The time displayed in the chart below is measured from the moment the box is picked up at the entry point to the moment it is placed in its allocated storage place.

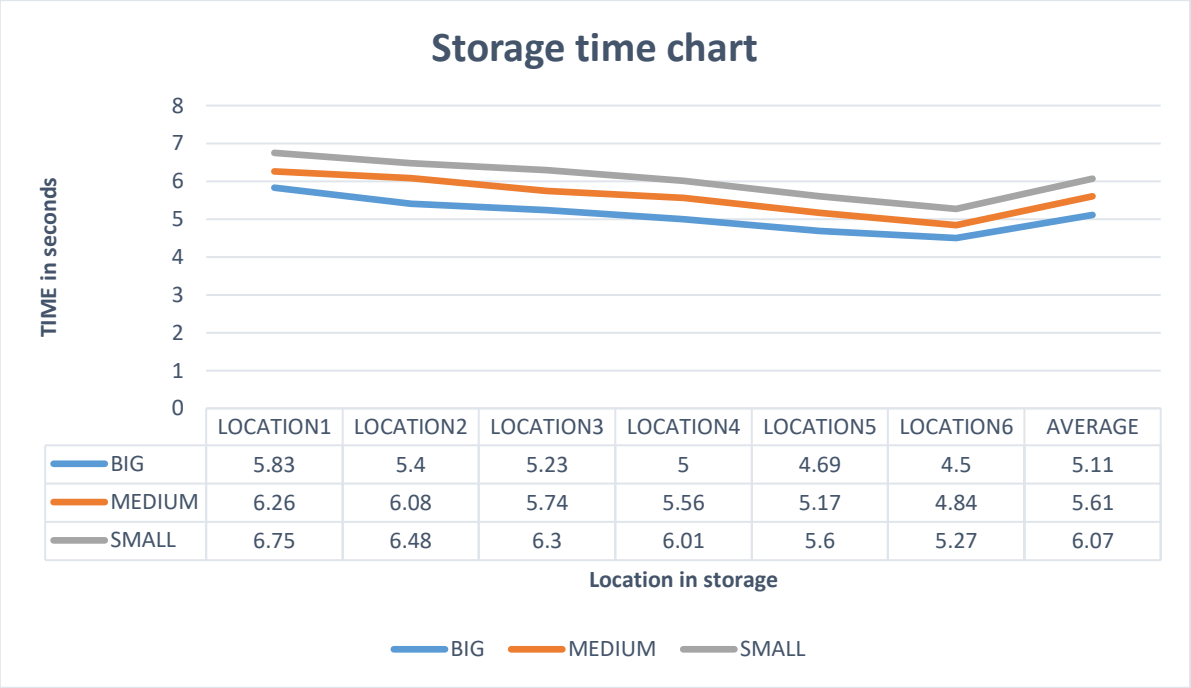


Figure 4.13 storage time chart.

The storage system has been proven to be able to pack 18 boxes in 2 minutes and 15seconds without interruption and unpack the entire stack in 1 minute and 50seconds. This proved to be efficient and will make a large impact in a bigger scale warehouse system.

4.4 System's operation

This section of this chapter focuses on the analysis of the tests and results obtained when combining the entire system as shown in Figure 1.1 of the introduction chapter. Each of parts of this system have been studied in the literature review (chapter 2), developed, programmed or built in the methodology (chapter 3) and finally test in the previous sections of this chapter. The rest of this chapter now will focus on testing the main function of the system such as:

- The negotiation and sharing of resources and
- The execution all tasks loaded in the assembly lines

4.4.1 Test

The test discussed in this are used to determine whether the system developed in this study if fully functional and meet all the different criteria and functions described in the first chapter of this research. Three forms of test are run and described in the coming section of this unit.

4.4.1.1 *Task execution without temporarily priority shift*

In this test, the assembly lines are loaded with lists of tasks that they have to execute while negotiating and sharing the usage of the resources. the aim of this trial is to see how long it will take to the system to execute a certain amount of tasks by sharing the resources based on this following rules

- Compare the level of priority of the tasks being started
- The task with the highest priority gets executed while the other waits
- The task waiting is compared to the next task on the assembly line with the previous high priority task.
- The task with the highest priority gets executed while the other waits

These steps will repeat until all the tasks of both assembly lines are completed.

4.4.1.2 *Task execution with temporarily priority shift*

This test is similar to the test in 4.4.1.1 above, the main difference is that we enable the temporary control of the resources. This allows the task with lower priority to be executed almost at the same time as the one that has the higher priority on the other assembly line. Table 1.1 in Chapter 1 on page 3 shows that each task has a number of parts that it needs and each part has an assembling and collection time. Those time are used to enable the temporary control function.

Function's example:

To better understand this function, the author makes use of this example where 2 assembly line are simultaneously executing task while sharing resources.

Table 4.2 example

ASSEMBLY LINE 1		ASSEMBLY LINES 2	
JOB1	PART2	JOB2	PART1
	PART3		PART2
	PART3		PART2
	PART4		PART3

The first assembly line is starting job1 while the second is starting job2. Using Table 1.1 in chapter 1 as reference, we know the following factors:

- Job1 has a higher execution priority than Job2
- Job1 requires part2, 2x part3, and part4 to be completed.

- Job2 requires part1, 2x part2, and part3 to be completed.

At the start of the tasks execution, the following execution sequence will be followed repeatedly until all the required parts are collected. The system is at the state where assembly line 2 is waiting for the task is assembly line1 to be completed and assembly line 1 is in full control of the resources.

Every time that a part is delivered to assembly line 1, a subroutine is run in assembly lines that compared the assembly time of the collected part to the collection time of the part to be collected for assembly line 2. If the assembling time is greater than the collection time, assembly line 2 will temporarily take control of the resources long enough for the part that it need to be collect then priority will be restored back to assembly line1 while assembly line 2 will also be busy assembling a part.

The results related to this test are shown and explained in section 4.4.2.2 of this chapter below.

4.4.1.3 All task completed task sharing

For this test, one of the assembly lines is loaded with a list that contains only 1 task in it while the other is loaded with 7 task. The assembly line with the largest amount of tasks will share it remaining tasks at the moment that the other is done.

4.4.2 Results

In this section, the author shows and explains the results obtained from the tests explained in the previous paragraphs above. It has been divided into 3 sub-sections for better understanding of the purpose of the test and their results.

4.4.2.1 Task execution without temporary priority shift

Table 4.3 bellow, shows the list of task that was loaded to the assembly lines. The orange dotted line indicates the order in which they were executed. Refer to Table 1.1 in chapter 3 for priority level of each task on the Table 4.3 below.

Table 4.3 Task list

Master 1 Task List	Task Name	Part List	A_T	C_T	W_T	M1	Master 2 Task List	Task Nam	Part List	A_T	C_T	W_T	M2
	job_four	Part1	5	1	83	89		job_three	Part1	5	1	0	6
		Part1	5	1	89	95			Part1	5	1	6	12
		Part2	10	1	95	106			Part2	10	1	12	23
		Part2	10	1	106	117			Part3	15	1	23	39
	job_three	Part1	5	1	117	123		job_two	Part1	5	1	39	45
		Part1	5	1	123	129			Part2	10	1	45	56
		Part2	10	1	129	140			Part2	10	1	56	67
		Part3	15	1	140	156			Part3	15	1	67	83
	job_three	Part1	5	1	156	162		job_four	Part1	5	1	303	309
		Part1	5	1	162	168			Part1	5	1	309	315
		Part2	10	1	168	179			Part2	10	1	315	326
		Part3	15	1	179	195			Part2	10	1	326	337
	job_two	Part1	5	1	195	201		job_four	Part1	5	1	337	343
		Part2	10	1	201	212			Part1	5	1	343	349
		Part2	10	1	212	223			Part2	10	1	349	360
		Part3	15	1	223	239			Part2	10	1	360	371
	job_one	Part2	10	1	239	250		job_two	Part1	5	1	371	377
		Part3	15	1	250	266			Part2	10	1	377	388
		Part3	15	1	266	282			Part2	10	1	388	399
		Part4	20	1	282	303			Part3	15	1	399	415

Table 4.3 and Table 4.4 column abbreviation:

- A_T: *Assembling Time*

  This is the time that it takes to assemble a part.

- C_T: *Collection Time*

  This is regarded as the time that it take for the AGV to collect a part from the storeroom to the assembly line's location.

- W_T: *Wait Time*

  It is the time that each part as to wait before being collected.

- M1 & M2: the total time that it takes to complete a task or part.

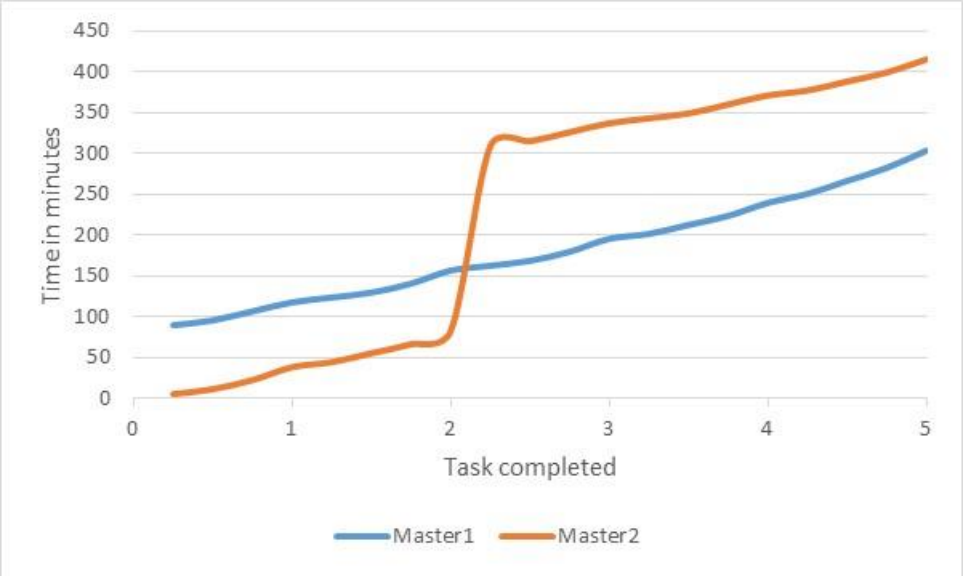


Figure 4.14 Master 1 & 2 Comparison

NB: *due to the fact that the life time of the battery on the AGV does not allow us to test the system from start to the end, the AGV and Storeroom processes have been simulated to get the results shown on this document. Therefore, all the collection times have been changed to 1 minute.*

Figure 4.14 above is a graph result of the comparison between the tasks execution on assembly line 1 and assembly line 2. This chart clearly shows the large difference in the operation of the 2 assembly lines and it is clearly seen that the system takes too long to execute all the tasks.

4.4.2.2 Task execution with temporary priority shift

The Table 4.4 Task list below is the list of tasks that were loaded to the assembly lines, it is the same table as the Table 4.3 in the previous point 4.4.2.1. The difference is that in this table the priority to use the AGV and storeroom shifts depending on the assembling and collection time of the parts on the list.

For example: the tasks to be executed are job_four on assembly line 1 and job_three on assembly line 2. The first test is to check the priority level where job_three has the higher priority. Then, every time that a part is collected to complete job_three, its assembling time is compared to the collection of the part in job_four. If that assembling time is greater than the collection time, then the AGV will collect that part for job_four on assembly line 1 while assembly line 2 is busy assembling the part1.

Table 4.4 Task list

Master 1 Task List	Task Name	Part List	A_T	C_T	W_T	M1
	job_four	Part1	5	1	1	7
		Part1	5	1	7	13
		Part2	10	1	13	24
		Part2	10	1	24	35
	job_three	Part1	5	1	39	45
		Part1	5	1	45	51
		Part2	10	1	51	62
		Part3	15	1	62	78
	job_three	Part1	5	1	83	89
		Part1	5	1	89	95
		Part2	10	1	95	106
		Part3	15	1	106	122
	job_two	Part1	5	1	122	128
		Part2	10	1	128	139
		Part2	10	1	139	150
		Part3	15	1	150	166
	job_one	Part2	10	1	166	177
		Part3	15	1	177	193
		Part3	15	1	193	209
		Part4	20	1	209	230

Master 2 Task List	Task Name	Part List	A_T	C_T	W_T	M2
	job_three	Part1	5	1	0	6
		Part1	5	1	6	12
		Part2	10	1	12	23
		Part3	15	1	23	39
	job_two	Part1	5	1	39	45
		Part2	10	1	45	56
		Part2	10	1	56	67
		Part3	15	1	67	83
	job_four	Part1	5	1	84	90
		Part1	5	1	90	96
		Part2	10	1	96	107
		Part2	10	1	107	118
	job_four	Part1	5	1	118	124
		Part1	5	1	124	130
		Part2	10	1	130	141
		Part2	10	1	141	152
	job_two	Part1	5	1	152	158
		Part2	10	1	158	169
		Part2	10	1	169	180
		Part3	15	1	180	196

This process greatly improved the overall time that the system takes to complete all the tasks loaded on the 2 assembly lines. The Figure 4.15 below can be compared to the Figure 4.14 in the previous unit 4.4.2.1 above to see how greatly the system has improved. The system takes almost half the time it took before to complete the same amount of tasks has shown in Figure 4.15 below.

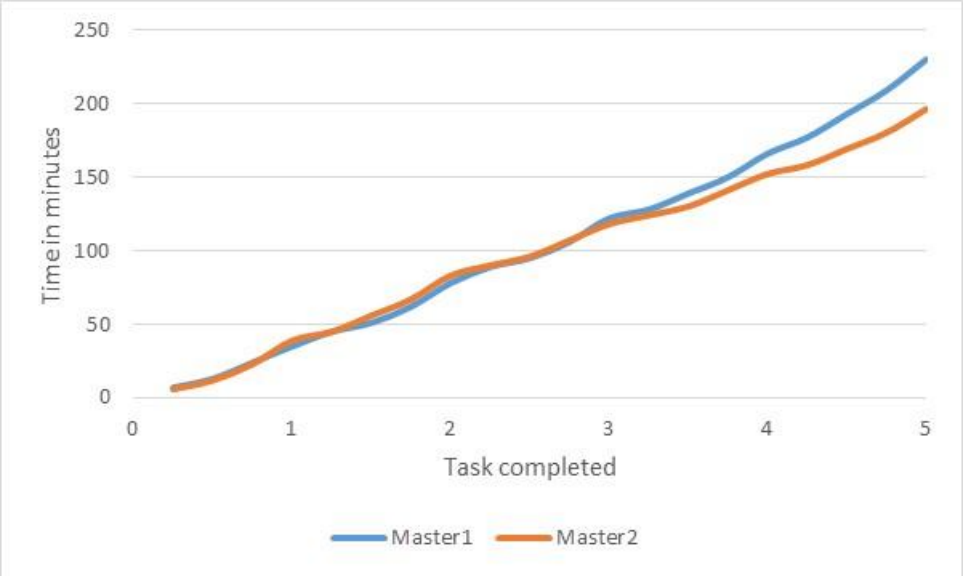


Figure 4.15 Master 1 & 2 Comparison

4.4.2.3 All task completed task sharing

The list of task loaded to the assembly lines to perform this test and quickly get these results have been modified to reduce their time in the following manner:

- Part1 assembling time was changed from 5 to 2 minutes
- Part2 assembling time was changed from 10 to 3 minutes
- Part3 assembling time was changed from 15 to 5 minutes
- Part4 assembling time was changed from 20 to 7 minutes

With these changes, the results obtained are shown the assembly lines logs in the Appendixes Assembly line 1 log and Assembly line 2 log below. These logs are used to explain the results the next paragraphs.

4.4.2.3.1 Task starting

The task starting on assembly line 1 is shown in Line 3 of the Assembly line 1 log and on Line 5 of the Assembly line 2 log, we can also see the task starting on assembly line 2.

Assembly line 2 only has one task on its list and that task is completed on Line 50 of Assembly line 2 log and Line 49 of Assembly line 1 log.

Table 4.5 Task list before sharing

Assembly line 1 task list			Assembly line 2 task list	
Task name	status	Priority	Task name	status
Job_four	Completed	<	Job_three	Completed
Job_three				
Job_three				
Job_two				
Job_three				
Job_two				
Job_four				
Job_four				

4.4.2.3.2 Remaining task sharing

Assembly line 1 has 8 tasks loaded to its list. At the moment that assembly line 2 completes its first and only task, assembly line 1 would already be done with its first task too thanks to the temporary priority shift (refer to 4.4.2.2).

Line 51 of Assembly line 2 log is a message from assembly line 1 specifying the number of uncompleted tasks in its list. Line 51 of Assembly line 1 log shows the number of tasks that assembly line 2 has added to it list. The results of this process are shown in Table 4.6 below.

Table 4.6 Task list after sharing

Assembly line 1 task list				Assembly line 2 task list		
Task name	status	Log Line	Priority	Task name	status	Log Line
Job_four	Completed	46	<	Job_three	Completed	50
Job_three	Completed	94	<	Job_two	Completed	101
Job_three	Completed	143	>	Job_four	Completed	141
Job_two	Completed	193	>	Job_four	Completed	186
Job_three	Completed	203				

this conclude the series of test and results used to verify the system's operation and the next chapter will elaborate the author's final conclusion, limitation and possible future work related to this study.

5 CONCLUSION

The principal objective of this study was to develop an intelligent single worker utilization for component retrieval for multiple stationary assembly lines. To evaluate such an intelligent system, a small manufacturing control environment consisting of 2 assembly lines, an AGV, and a storeroom needed to be developed. The assembly lines are also referred to in this document as the assembly lines, the AGV as the worker, and the combination of the AGV and storeroom is referred to as the resources. The separate system's component mention above should be autonomous and independent from each other. The 2 assembly lines share information and negotiate on the usage of resources over a network link. They should be able to perform all the task assigned to them as effectively and as efficiently as possible.

These objectives were met throughout this document, starting with the literature review study done by the author in chapter 2. He researched manufacturing system's components such as assembly lines, AGVs, and automation in storage systems. These researches were later on used in the development of the system in chapter 3. The methodology chapter is divided into 3 main parts that consist of the subsystem of this project. The first step to successfully develop these subsystems was the creation of a network that will allow them to constantly communicate with each other. Next was the development of the program algorithm that controls the assembly line. This algorithm utilizes a state machine to control the assembly line's functions such as the task sharing process, the priority level check, and the task execution. Then was the implementation of the line following AGV. The AGV used 4 digital sensors to follow a black line on a white surface. The middle 2 sensors were used for line following, and the extreme 2 sensors the count the check points placed next to the followed line. This allowed the AGV to easily follow multiple paths. And finally was the development of an automated storage and retrieval system that is use in the storeroom. It used RFID to identify, store, and retrieve parts from the storeroom according to their specifications. Each one of the sub-systems were individually tested in chapter 4 to verify their main operations. Afterwards, they were combined to form the main system. The last 2 units in chapter 4 were dedicated to test and verify whether the final system meets the objective set at the start of this study. The author first tested the communication between the separate components, then the task sharing process, after that, the priority check process, and finally, the task execution process.

The results obtained after the testing sequences were found satisfactory, therefore, the proposed project in this study was develop successfully.

The project offers the following contribution:

- Autonomy

  The manufacturing system in this project is composed of individual sub-system that are capable of running on their own. This could allow the system manager to easily maintain or fix the system without interrupting other system's components.

- Task sharing and negotiation

  The algorithm developed for the resources usage negotiations and tasks sharing has been proven to be very efficient, therefore, it can be use in a larger manufacturing system with more AGV and Assembly lines.

- AGV guidance system

  In chapter 3 of this document, the author developed a guidance system that allowed the AGV to follow multiple paths using only 4 digital line following sensors. This model was implemented to reduce the number of digital sensor used to follow multiple paths.

- Automated storage and retrieval system

  This has been used as part of the storeroom and provides an effective method to sort product according to their characteristics using RFID system.

The project developed in this study was done to test the working principles of a manufacturing system consisting of 2 assembly lines, an AGV, and a storeroom. This system, as illustrated throughout this document, was limited to the simulation of the time that it took for a task to be completed.

For future work, the author would like to test this system in a real manufacturing environment where all the data displayed in Table 1.1 will be tested and recorded to analyse the system's operations in real time and condition.

An automated charging system will have to be developed for the AGV to charge while it is waiting for commands from the assembly lines. This charging system will have to be place at the point where the AGV stops (collection point at the storeroom and delivery point at the assembly lines).

The addition of a monitoring system will also be beneficial to the user to monitor the progress of all the operations on the assembly lines, the quantity of parts left in the storeroom, and the battery level of the AGV. With this system, the manager will know whether the storeroom in resupply, or the AGV battery needs charging, or even add/remove tasks from the assembly lines while they are busy.

To conclude, modern manufacturing industries make use of automated systems to improve their productivity, save time, and to gain a certain advantage in the market over their competitor. These systems are often comprised of assembly line, a network of AGV that transport materials in the factory, a storage area where completed or uncompleted products are store, and a central control system that coordinate the operation of all the other. As great as these automated systems are, researches are still being done to improve and make them more efficient. The research conducted throughout this document aimed to enhance existing automated manufacturing systems. The author implemented a system that is made of two assembly lines, an AGV, and a storeroom. Where each of these parts are autonomously adept to function on their own without the need for a central control system. This allows the system to be maintained with ease when needed. The assembly lines utilize a programmed algorithm that sets the rules for information sharing, data backup, and negotiations for AGV and storeroom usage. This project has for advantage that it greatly reduced the overall time that a typical manufacturing system takes to complete the assembling of a given amount of products while efficiently sharing the one AGV and storeroom. The author believes that the implementation of this study to the manufacturing industries will significantly improve their productivity, revenues, save them time, effectively manage their storage space, and save them money by reducing the number of AGV needed per factories.

6 REFERENCES

[1] R. Singh, *Introduction to Basic Manufacturing Process and Workshop Technology*. New Age International, 2006.

[2] J. X. Wang, *Cellular Manufacturing: Mitigating Risk and Uncertainty*. CRC Press, 2015.

[3] J.-C. Spender and H. Kijne, *Scientific Management: Frederick Winslow Taylor's Gift to the World?* Springer Science & Business Media, 2012.

[4] S. Akpinar and A. Baykasoglu, "Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model," *Journal of manufacturing systems,* vol. 33, no. 1, pp. 177-187, 2014.

[5] M. F. F. Rashid, W. Hutabarat, and A. Tiwari, "A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches," *The International Journal of Advanced Manufacturing Technology,* vol. 59, no. 1-4, pp. 335-349, 2012.

[6] M. Vilà and J. Pereira, "A branch-and-bound algorithm for assembly line worker assignment and balancing problems," *Computers & Operations Research,* vol. 44, pp. 105-114, 2014.

[7] Ö. Mutlu, O. Polat, and A. A. Supciller, "An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II," *Computers & Operations Research,* vol. 40, no. 1, pp. 418-426, 2013.

[8] A. Yolmeh and F. Kianfar, "An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times," *Computers & Industrial Engineering,* vol. 62, no. 4, pp. 936-945, 2012.

[9] A. Roshani, P. Fattahi, A. Roshani, M. Salehi, and A. Roshani, "Cost-oriented two-sided assembly line balancing problem: A simulated annealing approach," *International Journal of Computer Integrated Manufacturing,* vol. 25, no. 8, pp. 689-715, 2012.

[10] J. Rada-Vilela, M. Chica, Ó. Cordón, and S. Damas, "A comparative study of multi-objective ant colony optimization algorithms for the time and space assembly line balancing problem," *Applied Soft Computing,* vol. 13, no. 11, pp. 4370-4382, 2013.

[11] H. Bidgoli, *The internet encyclopedia*. John Wiley & Sons, 2004.

[12] S. Cass, "The top 10 programming languages Spectrums 2014 ranking [DataFlow]," *Spectrum, IEEE,* vol. 51, no. 7, pp. 68-68, 2014.

[13] A. Hejlsberg, S. Wiltamuth, and P. Golde, *C# language specification*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[14] J. Gosling, *The Java language specification*. Addison-Wesley Professional, 2000.

[15] M. F. Sanner, "Python: a programming language for software integration and development," *J Mol Graph Model,* vol. 17, no. 1, pp. 57-61, 1999.

[16] C. Jackson, *Learning to Program Using Python*. CreateSpace, an Amazon Company, 2011.

[17] B. Smith, *Object-Oriented Programming*. Springer, 2015.

[18] E. K. a. I. Krivy, "Object-Oriented Simulation of systems with sophisticated control," *International Journal of General Systems,* pp. 313–343, 2011.

[19] J. Niemann, "Development of a reconfigurable assembly system with enhanced control capabilities and virtual commissioning," Bloemfontein: Central University of Technology, Free State, 2013.

[20] P. Rao, *Cad/Cam: Principles & Application* (Mechanical engineering). the McGraw-Hill Companies, 2006.

[21] S. Automation. (2014, 29 June). *AGV Basics* [Online]. Available: http://www.agvsystems.com/agvs-basics/basics-agvs/.

[22] R. A. P/L, "Industrial AGV's," AGVs, Ed., ed. Australia, 2015.

[23] A. Alavudeen and N. Venkateshwaran, *Computer integrated manufacturing*. PHI Learning Pvt. Ltd., 2008.

[24] Roboteq. (2014, 19 june). *Precision Magnetic Track Following Sensor with Optional Gyrosope* [Online]. Available: http://www.roboteq.com/index.php/docman/magsensor-documents-and-files/mgs-documents-1/mgs-datasheets-1/37-mgs1600-datasheet/file.

[25] V. supplies. (2007, 19 june). *Opto-electronic sensors* [Online]. Available: https://vision-supplies.com/Content/pdf/pdf_en_soex_en.pdf.
[26] V. Aggarwal. (2013, 13 october). *maxEmbedded a guide to robotics, embedded electronics and computer vision* [Online]. Available: http://maxembedded.com/2013/08/how-to-build-an-ir-sensor/.
[27] N. Instruments. (2013, 19 JUNE). *What is LabVIEW?* [Online]. Available: http://www.ni.com/newsletter/51141/en/.
[28] J. Travis. (19 june). *Introduction to Graphical Programming with LabVIEW* [Online]. Available: http://www.informit.com/articles/article.aspx?p=662895&seqNum=3.
[29] M. Yu, "Enhancing Storeroom Performance by Efficient Order Picking," 2008.
[30] R. Bagve, V. Kumbhar, M. D. Bhat, S. V. Verleker, and J. Fernandes, "Automatic Packing Machine & Material Handling using Programmable Logic Controller (PLC)," *International Journal for Innovative Research in Science and Technology,* vol. 2, no. 10, pp. 24-29, 2016.
[31] J. Marshall, "Programmable Logic Controllers: Essential and Affordable," presented at the 120th ASEE Annual Conference and Exposition, Atlanta, 23-26 June 2013, 2013.
[32] M. i. systems. (2007, 4 Febuary). *Making Information Work* [Online]. Available: http://www.machine-information-systems.com/PLC.html.
[33] M. Morshed, "Effective protocols for privacy and security in RFID systems applications," Staffordshire University, 2012.
[34] U. A. Bakshi and A. V. Bakshi, *Electrical Machines And Instruments*. Technical Publications, 2007, pp. 366-384.
[35] N. t. Corporation. (2014, 4 Febuary). *The Step Motor and Stepper Motor Controller* [Online]. Available: http://www.nmbtc.com/step-motors/engineering/basic-introduction-of-step-motors/.
[36] C. M. Kozierok, "TCP/IP Transport layer protocol," in *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*: No Starch Press, 2005, pp. 687-709.
[37] M. J. D. a. K. L. C. D. B. Makofske, *TCP/IP Sockets in C# Practical Guide for Programmers*. San Francisco: Morgan Kaufmann Publishers, 2004.
[38] M. Rouse. (1 October). *TCP/IP (Transmission Control Protocol/Internet Protocol) definition* [Online]. Available: http://searchnetworking.techtarget.com/definition/TCP-IP.
[39] N. Instruments. (30 September). *Basic TCP/IP Communication in LabVIEW* [Online]. Available: http://www.ni.com/white-paper/2710/en/.
[40] N. Instruments. (2015, 30 September). *TCP VI and Functions* [Online]. Available: http://zone.ni.com/reference/en-XX/help/371361M-01/lvcomm/tcp_vi_descriptions/#examples.
[41] H. i. Kalkan. (20011, 06 November). *A Complete TCP Server/Client Communication and RMI Framework in C# .NET - Implementation* [Online]. Available: http://www.codeproject.com/Articles/155282/TCP-Server-Client-Communication-Implementation#ScsWhatIsTcp.
[42] N. Instruments. (2012, 19 june). *NI LabVIEW Robotics Starter Kit for Prototyping* [Online]. Available: http://sine.ni.com/nips/cds/print/p/lang/en/nid/208010.
[43] N. Instruments. (2012, 19 june). *NI LabVIEW Robotics Starter Kit Robotics Platform for Teaching, Research, and Prototyping* [Online]. Available: http://sine.ni.com/ds/app/doc/p/id/ds-217/lang/en.
[44] N. Instruments. (2015, 19 May). *Overview of the LabVIEW Robotics Module* [Online]. Available: http://www.ni.com/white-paper/11564/en/.
[45] N. Instruments. (2013, 19 May). *LabVIEW Delivers Embedded Programming to New NI Single-Board RIO Platform* [Online]. Available: http://www.ni.com/newsletter/50452/en/.
[46] N. Instruments. (2014, September 2015). *NI sbRIO-9632/9632XT* [Online]. Available: Available: http://zone.ni.com/reference/en-XX/help/373197D-01/sbriohelp/ni_9632/.

7 Appendixes

7.1 Assembly line 1 log

Line 0: Assembly line1 : is connected
Line 1: AGV online
Line 2: Assembly line2 : is connected
Line 3: Tasks list received from : Assembly line2
Line 4: Assembly line2 task is Starting
Line 5: Job_three Task
Line 6: Assembly line2 task has higher priority
Line 7: Assembly line2 started task: Job_three
Line 8: Estimated waiting time: 16min
Line 9: AGV busy collecting Part1 for: Assembly line2
Line 10: Part delivered to Assembly line2
Line 11: 02:35 PM : Assembly line2 is Assembling Part1 time for completion: 2 Min
Line 12: AGV@StoreRoom
Line 13: AGV busy collecting Part1 for: Assembly line1
Line 14: Part delivered to Assembly line1
Line 15: AGV@StoreRoom
Line 16: 02:37 PM : Assembly line2 : Part1 Assembling process completed
Line 17: Assembly line2 is requesting the next Part on the list
Line 18: AGV busy collecting Part1 for: Assembly line2
Line 19: Part delivered to Assembly line2
Line 20: 02:37 PM : Assembly line2 is Assembling Part1 time for completion: 2 Min
Line 21: 2:37 PM : Assembly line1 : Part1 Assembling process completed
Line 22: AGV@StoreRoom
Line 23: AGV busy collecting Part1 for: Assembly line1
Line 24: Part delivered to Assembly line1
Line 25: AGV@StoreRoom
Line 26: 02:39 PM : Assembly line2 : Part1 Assembling process completed
Line 27: Assembly line2 is requesting the next Part on the list
Line 28: AGV busy collecting Part2 for: Assembly line2
Line 29: Part delivered to Assembly line2
Line 30: 02:39 PM : Assembly line2 is Assembling Part2 time for completion: 3 Min
Line 31: 2:39 PM : Assembly line1 : Part1 Assembling process completed
Line 32: AGV@StoreRoom
Line 33: AGV busy collecting Part2 for: Assembly line1
Line 34: Part delivered to Assembly line1
Line 35: AGV@StoreRoom
Line 36: 02:42 PM : Assembly line2 : Part2 Assembling process completed
Line 37: Assembly line2 is requesting the next Part on the list
Line 38: AGV busy collecting Part3 for: Assembly line2
Line 39: Part delivered to Assembly line2
Line 40: 02:42 PM : Assembly line2 is Assembling Part3 time for completion: 5 Min
Line 41: 2:42 PM : Assembly line1 : Part2 Assembling process completed
Line 42: AGV@StoreRoom
Line 43: AGV busy collecting Part2 for: Assembly line1
Line 44: Part delivered to Assembly line1
Line 45: AGV@StoreRoom
Line 46: 2:45 PM : Assembly line1 : Part2 Assembling process completed
Line 47: 02:47 PM : Assembly line2 : Part3 Assembling process completed
Line 48: Assembly line2 Job_three all parts collected

Line 49: Assembly line2 : All tasks Completed
Line 50: Assembly line2 is free
Line 51: Assembly line2 has added 3 tasks form Assembly line1 to its list
Line 52: Assembly line2 task is Starting
Line 53: Job_two Task
Line 54: Assembly line2 task has higher priority
Line 55: Assembly line2 started task: Job_two
Line 56: Estimated waiting time: 17min
Line 57: AGV busy collecting Part1 for: Assembly line2
Line 58: Part delivered to Assembly line2
Line 59: 02:47 PM : Assembly line2 is Assembling Part1 time for completion: 2 Min
Line 60: AGV@StoreRoom
Line 61: AGV busy collecting Part1 for: Assembly line1
Line 62: Part delivered to Assembly line1
Line 63: AGV@StoreRoom
Line 64: 02:49 PM : Assembly line2 : Part1 Assembling process completed
Line 65: Assembly line2 is requesting the next Part on the list
Line 66: AGV busy collecting Part2 for: Assembly line2
Line 67: Part delivered to Assembly line2
Line 68: 02:49 PM : Assembly line2 is Assembling Part2 time for completion: 3 Min
Line 69: 2:49 PM : Assembly line1 : Part1 Assembling process completed
Line 70: AGV@StoreRoom
Line 71: AGV busy collecting Part1 for: Assembly line1
Line 72: Part delivered to Assembly line1
Line 73: AGV@StoreRoom
Line 74: 2:51 PM : Assembly line1 : Part1 Assembling process completed
Line 75: 02:52 PM : Assembly line2 : Part2 Assembling process completed
Line 76: Assembly line2 is requesting the next Part on the list
Line 77: AGV busy collecting Part2 for: Assembly line2
Line 78: Part delivered to Assembly line2
Line 79: 02:52 PM : Assembly line2 is Assembling Part2 time for completion: 3 Min
Line 80: AGV@StoreRoom
Line 81: AGV busy collecting Part2 for: Assembly line1
Line 82: Part delivered to Assembly line1
Line 83: AGV@StoreRoom
Line 84: 02:55 PM : Assembly line2 : Part2 Assembling process completed
Line 85: Assembly line2 is requesting the next Part on the list
Line 86: AGV busy collecting Part3 for: Assembly line2
Line 87: Part delivered to Assembly line2
Line 88: 2:55 PM : Assembly line1 : Part2 Assembling process completed
Line 89: 02:55 PM : Assembly line2 is Assembling Part3 time for completion: 5 Min
Line 90: AGV@StoreRoom
Line 91: AGV busy collecting Part3 for: Assembly line1
Line 92: Part delivered to Assembly line1
Line 93: AGV@StoreRoom
Line 94: 3:00 PM : Assembly line1 : Part3 Assembling process completed
Line 95: 03:00 PM : Assembly line2 : Part3 Assembling process completed
Line 96: Assembly line2 Job_two all parts collected
Line 97: Assembly line2 task : Job_two was Completed
Line 98: Assembly line2 task is Starting
Line 99: Job_four Task
Line 100: Assembly line2 task has lower priority
Line 101: AGV busy collecting Part1 for: Assembly line1
Line 102: Part delivered to Assembly line1

Line 103: Assembly line2 is in control of the AGV temporarely
Line 104: AGV@StoreRoom
Line 105: AGV busy collecting Part1 for: Assembly line2
Line 106: Part delivered to Assembly line2
Line 107: 03:00 PM : Assembly line2 is Assembling Part1 time for completion: 2 Min
Line 108: AGV@StoreRoom
Line 109: 3:02 PM : Assembly line1 : Part1 Assembling process completed
Line 110: Assembly line2 is waiting
Line 111: AGV busy collecting Part1 for: Assembly line1
Line 112: Part delivered to Assembly line1
Line 113: Assembly line2 is in control of the AGV temporarely
Line 114: AGV@StoreRoom
Line 115: AGV busy collecting Part1 for: Assembly line2
Line 116: Part delivered to Assembly line2
Line 117: 03:02 PM : Assembly line2 is Assembling Part1 time for completion: 2 Min
Line 118: AGV@StoreRoom
Line 119: 3:04 PM : Assembly line1 : Part1 Assembling process completed
Line 120: Assembly line2 is waiting
Line 121: AGV busy collecting Part2 for: Assembly line1
Line 122: Part delivered to Assembly line1
Line 123: Assembly line2 is in control of the AGV temporarely
Line 124: AGV@StoreRoom
Line 125: AGV busy collecting Part2 for: Assembly line2
Line 126: Part delivered to Assembly line2
Line 127: 03:04 PM : Assembly line2 is Assembling Part2 time for completion: 3 Min
Line 128: AGV@StoreRoom
Line 129: 3:07 PM : Assembly line1 : Part2 Assembling process completed
Line 130: Assembly line2 is waiting
Line 131: AGV busy collecting Part3 for: Assembly line1
Line 132: Part delivered to Assembly line1
Line 133: Assembly line2 is in control of the AGV temporarely
Line 134: AGV@StoreRoom
Line 135: AGV busy collecting Part2 for: Assembly line2
Line 136: Part delivered to Assembly line2
Line 137: 03:07 PM : Assembly line2 is Assembling Part2 time for completion: 3 Min
Line 138: AGV@StoreRoom
Line 139: 03:10 PM : Assembly line2 : Part2 Assembling process completed
Line 140: Assembly line2 has completed Job_four
Line 141: 3:12 PM : Assembly line1 : Part3 Assembling process completed
Line 142: Assembly line2 waiting is deActivated
Line 143: Assembly line1 task : Job_three was Completed
Line 144: Assembly line2 task is Starting
Line 145: Job_four Task
Line 146: Assembly line2 task has lower priority
Line 147: AGV busy collecting Part1 for: Assembly line1
Line 148: Part delivered to Assembly line1
Line 149: Assembly line2 is in control of the AGV temporarely
Line 150: AGV@StoreRoom
Line 151: AGV busy collecting Part1 for: Assembly line2
Line 152: Part delivered to Assembly line2
Line 153: 03:12 PM : Assembly line2 is Assembling Part1 time for completion: 2 Min
Line 154: AGV@StoreRoom
Line 155: 3:14 PM : Assembly line1 : Part1 Assembling process completed
Line 156: Assembly line2 is waiting

Line 157: AGV busy collecting Part2 for: Assembly line1
Line 158: Part delivered to Assembly line1
Line 159: Assembly line2 is in control of the AGV temporarely
Line 160: AGV@StoreRoom
Line 161: AGV busy collecting Part1 for: Assembly line2
Line 162: Part delivered to Assembly line2
Line 163: 03:14 PM : Assembly line2 is Assembling Part1 time for completion: 2 Min
Line 164: AGV@StoreRoom
Line 165: 03:16 PM : Assembly line2 : Part1 Assembling process completed
Line 166: Assembly line2 temporary AGV control is deActivated
Line 167: 3:17 PM : Assembly line1 : Part2 Assembling process completed
Line 168: Assembly line2 is waiting
Line 169: AGV busy collecting Part2 for: Assembly line1
Line 170: Part delivered to Assembly line1
Line 171: Assembly line2 is in control of the AGV temporarely
Line 172: AGV@StoreRoom
Line 173: AGV busy collecting Part2 for: Assembly line2
Line 174: Part delivered to Assembly line2
Line 175: 03:17 PM : Assembly line2 is Assembling Part2 time for completion: 3 Min
Line 176: AGV@StoreRoom
Line 177: 3:20 PM : Assembly line1 : Part2 Assembling process completed
Line 178: Assembly line2 is waiting
Line 179: AGV busy collecting Part3 for: Assembly line1
Line 180: Part delivered to Assembly line1
Line 181: Assembly line2 is in control of the AGV temporarely
Line 182: AGV@StoreRoom
Line 183: AGV busy collecting Part2 for: Assembly line2
Line 184: Part delivered to Assembly line2
Line 185: 03:20 PM : Assembly line2 is Assembling Part2 time for completion: 3 Min
Line 186: AGV@StoreRoom
Line 187: 03:23 PM : Assembly line2 : Part2 Assembling process completed
Line 188: Assembly line2 has completed Job_four
Line 189: Assembly line2 : All tasks Completed
Line 190: Assembly line2 is free
Line 191: 3:25 PM : Assembly line1 : Part3 Assembling process completed
Line 192: Assembly line2 waiting for new task to be added to its list
Line 193: Assembly line1 task : Job_two was Completed
Line 194: AGV busy collecting Part1 for: Assembly line1
Line 195: Part delivered to Assembly line1
Line 196: Assembly line2 is free
Line 197: AGV@StoreRoom
Line 198: 3:27 PM : Assembly line1 : Part1 Assembling process completed
Line 199: AGV busy collecting Part2 for: Assembly line1
Line 200: Part delivered to Assembly line1
Line 201: AGV@StoreRoom
Line 202: 3:32 PM : Assembly line1 : Part3 Assembling process completed
Line 203: Assembly line1 : All tasks Completed
Line 204: Assembly line2 waiting for new task to be added to its list
Line 205: All task in the system have been completed
Line 206: Assembly line2 disconnected
Line 207:

7.2 Assembly line 2 log

Line 0: Assembly line2 : is connected
Line 1: Tasks list received from : Assembly line1
Line 2: Assembly line1 task is Starting
Line 3: Job_four Task
Line 4: Assembly line1 task has lower priority
Line 5: AGV busy collecting Part1 for: Assembly line2
Line 6: Part delivered to Assembly line2
Line 7: Assembly line1 is in control of the AGV temporarely
Line 8: AGV@StoreRoom
Line 9: AGV busy collecting Part1 for: Assembly line1
Line 10: Part delivered to Assembly line1
Line 11: 2:35 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 12: AGV@StoreRoom
Line 13: 02:37 PM : Assembly line2 : Part1 Assembling process completed
Line 14: Assembly line1 is waiting
Line 15: AGV busy collecting Part1 for: Assembly line2
Line 16: Part delivered to Assembly line2
Line 17: Assembly line1 is in control of the AGV temporarely
Line 18: 2:37 PM : Assembly line1 : Part1 Assembling process completed
Line 19: AGV@StoreRoom
Line 20: AGV busy collecting Part1 for: Assembly line1
Line 21: Part delivered to Assembly line1
Line 22: 2:37 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 23: AGV@StoreRoom
Line 24: 02:39 PM : Assembly line2 : Part1 Assembling process completed
Line 25: Assembly line1 is waiting
Line 26: AGV busy collecting Part2 for: Assembly line2
Line 27: Part delivered to Assembly line2
Line 28: Assembly line1 is in control of the AGV temporarely
Line 29: 2:39 PM : Assembly line1 : Part1 Assembling process completed
Line 30: AGV@StoreRoom
Line 31: AGV busy collecting Part2 for: Assembly line1
Line 32: Part delivered to Assembly line1
Line 33: 2:39 PM : Assembly line1 is Assembling Part2 time for completion: 3 Min
Line 34: AGV@StoreRoom
Line 35: 02:42 PM : Assembly line2 : Part2 Assembling process completed
Line 36: Assembly line1 is waiting
Line 37: AGV busy collecting Part3 for: Assembly line2
Line 38: Part delivered to Assembly line2
Line 39: Assembly line1 is in control of the AGV temporarely
Line 40: 2:42 PM : Assembly line1 : Part2 Assembling process completed
Line 41: AGV@StoreRoom
Line 42: AGV busy collecting Part2 for: Assembly line1
Line 43: Part delivered to Assembly line1
Line 44: 2:42 PM : Assembly line1 is Assembling Part2 time for completion: 3 Min
Line 45: AGV@StoreRoom
Line 46: 2:45 PM : Assembly line1 : Part2 Assembling process completed
Line 47: Assembly line1 has completed Job_four
Line 48: 02:47 PM : Assembly line2 : Part3 Assembling process completed

Line 49: Assembly line1 waiting is deActivated
Line 50: Assembly line2 : All tasks Completed
Line 51: Assembly line1 has 7 tasks left
Line 52: Assembly line1 task is Starting
Line 53: Job_three Task
Line 54: Assembly line1 task has lower priority
Line 55: AGV busy collecting Part1 for: Assembly line2
Line 56: Part delivered to Assembly line2
Line 57: Assembly line1 is in control of the AGV temporarely
Line 58: AGV@StoreRoom
Line 59: AGV busy collecting Part1 for: Assembly line1
Line 60: Part delivered to Assembly line1
Line 61: 2:47 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 62: AGV@StoreRoom
Line 63: 02:49 PM : Assembly line2 : Part1 Assembling process completed
Line 64: Assembly line1 is waiting
Line 65: AGV busy collecting Part2 for: Assembly line2
Line 66: Part delivered to Assembly line2
Line 67: Assembly line1 is in control of the AGV temporarely
Line 68: 2:49 PM : Assembly line1 : Part1 Assembling process completed
Line 69: AGV@StoreRoom
Line 70: AGV busy collecting Part1 for: Assembly line1
Line 71: Part delivered to Assembly line1
Line 72: 2:49 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 73: AGV@StoreRoom
Line 74: 2:51 PM : Assembly line1 : Part1 Assembling process completed
Line 75: Assembly line1 temporary AGV control is deActivated
Line 76: 02:52 PM : Assembly line2 : Part2 Assembling process completed
Line 77: Assembly line1 is waiting
Line 78: AGV busy collecting Part2 for: Assembly line2
Line 79: Part delivered to Assembly line2
Line 80: Assembly line1 is in control of the AGV temporarely
Line 81: AGV@StoreRoom
Line 82: AGV busy collecting Part2 for: Assembly line1
Line 83: Part delivered to Assembly line1
Line 84: 2:52 PM : Assembly line1 is Assembling Part2 time for completion: 3 Min
Line 85: AGV@StoreRoom
Line 86: 02:55 PM : Assembly line2 : Part2 Assembling process completed
Line 87: Assembly line1 is waiting
Line 88: AGV busy collecting Part3 for: Assembly line2
Line 89: Part delivered to Assembly line2
Line 90: Assembly line1 is in control of the AGV temporarely
Line 91: 2:55 PM : Assembly line1 : Part2 Assembling process completed
Line 92: AGV@StoreRoom
Line 93: AGV busy collecting Part3 for: Assembly line1
Line 94: Part delivered to Assembly line1
Line 95: 2:55 PM : Assembly line1 is Assembling Part3 time for completion: 5 Min
Line 96: AGV@StoreRoom
Line 97: 3:00 PM : Assembly line1 : Part3 Assembling process completed
Line 98: Assembly line1 has completed Job_three
Line 99: 03:00 PM : Assembly line2 : Part3 Assembling process completed

Line 100: Assembly line1 waiting is deActivated
Line 101: Assembly line2 task : Job_two was Completed
Line 102: Assembly line1 task is Starting
Line 103: Job_three Task
Line 104: Assembly line1 task has higher priority
Line 105: Assembly line1 started task: Job_three
Line 106: Estimated waiting time: 16min
Line 107: AGV busy collecting Part1 for: Assembly line1
Line 108: Part delivered to Assembly line1
Line 109: 3:00 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 110: AGV@StoreRoom
Line 111: AGV busy collecting Part1 for: Assembly line2
Line 112: Part delivered to Assembly line2
Line 113: AGV@StoreRoom
Line 114: 3:02 PM : Assembly line1 : Part1 Assembling process completed
Line 115: Assembly line1 is requesting the next Part on the list
Line 116: AGV busy collecting Part1 for: Assembly line1
Line 117: Part delivered to Assembly line1
Line 118: 3:02 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 119: AGV@StoreRoom
Line 120: AGV busy collecting Part1 for: Assembly line2
Line 121: Part delivered to Assembly line2
Line 122: AGV@StoreRoom
Line 123: 3:04 PM : Assembly line1 : Part1 Assembling process completed
Line 124: Assembly line1 is requesting the next Part on the list
Line 125: AGV busy collecting Part2 for: Assembly line1
Line 126: Part delivered to Assembly line1
Line 127: 3:04 PM : Assembly line1 is Assembling Part2 time for completion: 3 Min
Line 128: AGV@StoreRoom
Line 129: AGV busy collecting Part2 for: Assembly line2
Line 130: Part delivered to Assembly line2
Line 131: AGV@StoreRoom
Line 132: 3:07 PM : Assembly line1 : Part2 Assembling process completed
Line 133: Assembly line1 is requesting the next Part on the list
Line 134: AGV busy collecting Part3 for: Assembly line1
Line 135: Part delivered to Assembly line1
Line 136: 3:07 PM : Assembly line1 is Assembling Part3 time for completion: 5 Min
Line 137: AGV@StoreRoom
Line 138: AGV busy collecting Part2 for: Assembly line2
Line 139: Part delivered to Assembly line2
Line 140: AGV@StoreRoom
Line 141: 03:10 PM : Assembly line2 : Part2 Assembling process completed
Line 142: 3:12 PM : Assembly line1 : Part3 Assembling process completed
Line 143: Assembly line1 Job_three all parts collected
Line 144: Assembly line1 task : Job_three was Completed
Line 145: Assembly line1 task is Starting
Line 146: Job_two Task
Line 147: Assembly line1 task has higher priority
Line 148: Assembly line1 started task: Job_two
Line 149: Estimated waiting time: 17min
Line 150: AGV busy collecting Part1 for: Assembly line1

Line 151: Part delivered to Assembly line1
Line 152: 3:12 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 153: AGV@StoreRoom
Line 154: AGV busy collecting Part1 for: Assembly line2
Line 155: Part delivered to Assembly line2
Line 156: AGV@StoreRoom
Line 157: 3:14 PM : Assembly line1 : Part1 Assembling process completed
Line 158: Assembly line1 is requesting the next Part on the list
Line 159: AGV busy collecting Part2 for: Assembly line1
Line 160: Part delivered to Assembly line1
Line 161: 3:14 PM : Assembly line1 is Assembling Part2 time for completion: 3 Min
Line 162: AGV@StoreRoom
Line 163: AGV busy collecting Part1 for: Assembly line2
Line 164: Part delivered to Assembly line2
Line 165: AGV@StoreRoom
Line 166: 03:16 PM : Assembly line2 : Part1 Assembling process completed
Line 167: 3:17 PM : Assembly line1 : Part2 Assembling process completed
Line 168: Assembly line1 is requesting the next Part on the list
Line 169: AGV busy collecting Part2 for: Assembly line1
Line 170: Part delivered to Assembly line1
Line 171: 3:17 PM : Assembly line1 is Assembling Part2 time for completion: 3 Min
Line 172: AGV@StoreRoom
Line 173: AGV busy collecting Part2 for: Assembly line2
Line 174: Part delivered to Assembly line2
Line 175: AGV@StoreRoom
Line 176: 3:20 PM : Assembly line1 : Part2 Assembling process completed
Line 177: Assembly line1 is requesting the next Part on the list
Line 178: AGV busy collecting Part3 for: Assembly line1
Line 179: Part delivered to Assembly line1
Line 180: 3:20 PM : Assembly line1 is Assembling Part3 time for completion: 5 Min
Line 181: AGV@StoreRoom
Line 182: AGV busy collecting Part2 for: Assembly line2
Line 183: Part delivered to Assembly line2
Line 184: AGV@StoreRoom
Line 185: 03:23 PM : Assembly line2 : Part2 Assembling process completed
Line 186: Assembly line2 : All tasks Completed
Line 187: 3:25 PM : Assembly line1 : Part3 Assembling process completed
Line 188: Assembly line1 Job_two all parts collected
Line 189: Assembly line1 task : Job_two was Completed
Line 190: AGV busy collecting Part1 for: Assembly line1
Line 191: Part delivered to Assembly line1
Line 192: 3:25 PM : Assembly line1 is Assembling Part1 time for completion: 2 Min
Line 193: AGV@StoreRoom
Line 194: 3:27 PM : Assembly line1 : Part1 Assembling process completed
Line 195: Assembly line1 is requesting the next Part on the list
Line 196: AGV busy collecting Part2 for: Assembly line1
Line 197: Part delivered to Assembly line1
Line 198: 3:27 PM : Assembly line1 is Assembling Part3 time for completion: 5 Min
Line 199: AGV@StoreRoom
Line 200: 3:32 PM : Assembly line1 : Part3 Assembling process completed
Line 201: Assembly line1 Job_three all parts collected

Line 202: Assembly line1 : All tasks Completed
Line 203: All task in the system have been completed
Line 204: Assembly line2 disconnected
Line 205: